nexacro platform 14 / 管理者ガイド

14.0.1.4000

発行先 | 日本ネクサウェブ株式会社
住所 | 〒101-0021　東京都千代田区外神田５-３-１秋葉原OSビル１０Ｆ
電話 | 03－5812－6358
ホームページ | www.nexaweb.co.jp

履歴

バージョン	変更日	内容
14.0.1.500.3	2016-09-01	DeviceAPI コードからcom.nexacro.permission関連コードを削除しました。 X-PUSH使用時アプリインストール問題により、関連コード削除しました。
14.0.1.500.3.1	2016-09-20	誤打訂正
14.0.1.700.1	2016-10-25	オプション使用オプション追加(-M, -MO)しました。 nexacrogenerator.exeの使用法使用例追加(-M, -MO, -TABORDER)しました。 nexacrocompressor.exe使用オプション追加(-Compile)しました。
14.0.1.700.2	2016-11-8	相対パスの項目に表記される内容のうち、nexacro studioとは違って表記された用語を修正しました。
14.0.1.700.3	2016-11-14	nexacrocompressor.exeの使用オプションの重複された項目を削除しました。
14.0.1.800.1	2016-12-01	インストールで、インストールされたモジュールリストを修正しました。 使用しないモジュールを削除（xmlpull-1.1.3.1.jar, xpp3_min-1.1.4c.jar, xstream-1.4.2.jar）しました。
14.0.1.800.2	2016-12-12	基本配布方法、主要モジュールの実行方法項目を追加しました。
14.0.1.900.1	2016-12-27	Runtime for iOS/Android 「ハイブリッド」という表現を削除しました。
14.0.1.900.2	2017-01-04	SampleLaunch.htmlとSampleStart.htmlの統合項目で、nexacro14_SetupEngine.cab内にLauncherが含まれているという内容を削除しました。
14.0.1.1200.1	2017-03-28	nexacrocompressor.exe 項目にIgnorecompressオプションを追加しました。
14.0.1.1200.3	2017-04-07	システム要件 「最新仕様」項目を追加しました。
14.0.1.1400	2017-05-30	checkversion 項目を追加しました。
14.0.1.1501.1	2017-07-07	アプリ開発と実行(iOS) 内容をXcode 7.3.1基準に修正しました。 システム要件項目のプラットフォームのサポート範囲を修正しました。 用語の変更：最新仕様→最高仕様
14.0.1.2200	2018-01-23	プロジェクト生成 項目にiPhone X関連の注意書きを追加しました。

バージョン	変更日	内容
14.0.1.2800.1	2018-10-22	Dataset > ColumnInfo エレメント配置時の注意項目を追加しました。
14.0.1.2800.2	2018-11-20	メッセージの設定 項目にメッセージを追加しました。
14.0.1.3000	2019-01-11	Notification機能（14.0.1.3000以降のバージョン）項目を追加しました。
14.0.1.3100.1	2019-04-08	nexacro-xeniインストール項目でJDKバージョンの表示エラーを修正しました。 JDK1.5以上のバージョンが必要です。
14.0.1.3200	2019-08-05	nexacro platformライブラリ設定 項目に64ビットライブラリに関する説明を追加しました。
14.0.1.3200.1	2019-09-16	AndroidManifest.xml説明にHTTP通信関連の注意事項を追加しました。
14.0.1.3300	2020-01-13	システム要件 項目のiOS最低仕様を9に変更しました。
14.0.1.3400	2021-09-08	カスタムフォントの設定項目を追加しました。
14.0.1.4000	2021-12-24	nexacro-xeniインストール項目にて、JDK（JRE）のバージョンを変更しました。また、jar（Java ARchive） 内のファイル一覧を削除しました。

目次

はじめに

本マニュアルは、nexacro platformの基本的な概要とnexacro platformで作られたアプリケーションを配布、実行段階で必要な作業について説明します。

対象読者

対象読者

配布や運用を担当する管理者が運営のために参照したり、アプリケーションを開発する開発者が詳細な動作を理解し、テストを実施するために参照することができます。

nexacro platformのプログラミング

nexacro platformのアプリケーション開発のためのプログラミングの詳細は、開発者ガイドを参照してください。

nexacro studio

nexacro studioインストールと構成、オプションに関する詳細については、nexacro studioガイドを参照してください。

その他の開発ツール

本社から提供されていないWebサーバーまたはその他の開発ツールについては、概略的な部分しか説明しています。各メーカーのマニュアルを別途参照してください。

マニュアルの構成

詳細項目の構成は以下のとおりです。

nexacro platformの紹介

nexacro platformについて簡単に説明します。

nexacro platformの概要

nexacro platformの基本的な構造と構成要素について説明します。

RUNTIME for iOS/Andriod

モバイル環境でのnexacro platformアプリケーションをパッケージ化してアプリで配布する内容を説明します。

参考

アプリケーションの配布や管理時に必要な付加的な内容を説明します。

マニュアルの表記法

本マニュアルは、読者の理解を高めるために特別な意味を持つ単語や文章を別途の表記法により表現しました。以下はその表記法についての説明です。

ノート、ヒント、注意は次のように提供されます。

ノートは、本体に簡単に追加する短い説明や参照、論評を提供するために使用します。.

ヒントは、ヘルプなどのヒントを提供するために使用します。

注意は、読者やユーザーの注意を喚起する文章を提供するために使用します。

第I部

概要

第1章

nexacro platformの紹介

業務に最適化されたシステム、必要な情報に簡単にアクセスすることができるシステム、直感的に実行できるシステム、そしてデータ入力作業のような面倒な作業に時間がかからず、必要な業務に集中できるシステムは、画期的な生産性の向上をもたらすことができます。このような変化は、システムのUI/ UX環境をどのように構築するかにかかっています。

UI/ UXの革新は、システムを単に見栄え良く作ることではなく、効率的に総所有コスト(TCO、Total Cost of Ownership)を削減し、投下資本利益率(ROI、Return On Investment)を高めながら、ユーザビリティも一緒に高めて経営者、システム管理者、そしてユーザーといったみんなの満足度を上げる必要があります。

nexacro platformは、機能、性能、デザイン要素が企業の要求水準に合わせて調和を成すように構成されていて、企業のために生まれたビジネスユーザーエクスペリエンスソリューションです。

1.1　企業でのUI/UX

RIA(Rich Internet Application)、そしてREA(Rich Enterprise Application)に至るまで、企業と個人のより豊かなユーザーエクスペリエンスを実現するためのWebアプリケーションの概念は続いて発展してきました。

C/S(Client Server)環境は処理速度は満足だったものの、インストール、配布、保守管理の難しさがあって、ウェブの登場によってこのような問題は解決されましたが、データ処理速度と制限されたUIといった限界を持つようになりました。それを克服するために、2002年Adobeと合併されたマクロメディアで使用していたRIAという用語が拡散されて全体の市場が拡大されました。RIAは一般ユーザーを対象にする広告やデザイン、アニメーションを強調して発展してきました。

時間が経つにつれて、企業用アプリケーション市場でも、より豊かで高度のユーザーインターフェイスを要求し始め、企業のためのRIAという意味のREAソリューションが企業向け市場で脚光を浴び始めています。ただし、REAは一般ユーザーを対象にしたRIAとは根本的なアプローチが異なっていました。徹底的に企業環境の分析と複雑な開発ニーズの理解、そして企業への高い理解度と数多くの構築経験に基づいて作られてきた独自的な領域です。

最近、"UX(User eXperience)"という用語が注目され、企業のビジネス環境に最適化されたUI/UXの実現を通じた企業の価値のあるビジネスビュー(VIEW)への関心が高まりました。このような新たな企業ユーザーのニーズに対応し、REAの概念を超えて拡張された概念でBUX(Business User eXperience：ビジネスユーザー経験)を定義しました。BUXは、直感的な業務環境、ユーザーの経験をベースにしたUIの実装を通じて、業務の生産性を高め、速やかな意思決定を可能にすることによって、何よりも企業顧客のビジネス価値を実現することに重心を置いています。

1.1.1 BUX: Business User eXperience

BUXは、以下の3つの統合によって実現されます。

一つ目は、プラットフォームの統合(Unified Platform)です。
さまざまなオペレーティングシステム、ブラウザ、およびデバイスはもちろん、それぞれ異なる画面サイズに対して1つのソースで実行され、最適化された結果が得られるOSMU(One Source Multi Use)を指向しています。

二つ目は、開発環境の統合(Unified Development Process)です。
同じ開発環境内でインハウス(In House)開発やSAPのようなUI高度化パッケージの開発作業を全て実行することができ、開発からテストおよび配布まで、ソフトウェア開発の全プロセスを統合できるようにする開発ツールである"nexacro studio"を提供することにより、開発生産性を向上させます。

三つ目は、データ中心の統合(Unified Data Frame)です。
定型、非定型データとOLTP(Online Transaction Processing)関連業務とビッグデータの可視化を含めたOLAP(Online Analysis Processing)関連業務の両方をサポートすることを意味します。これは、データ指向の企業用システムUI/ UXにおいて、業務効率性と生産性の向上に直結することであるので、特に重要な意味があります。

企業システムのUI/ UXの改善は投資コストの業務生産性向上と満足度向上効果が一番高い方法です。既存のシステムをほぼ変更することなく、大幅な改善効果を創出することができるためです。最小のコストを通じた最大の技術革新効果、そして、モバイル、クラウド、ビッグデータ技術を積極的に受け入れて、企業用UI/UXを統合的にサポートするという意志こそ、BUXプラットフォームであるnexacro platformが追求する方向です。

1.2 nexacro platform 14

nexacro platform 14は、BUXプラットフォームで、企業のさまざまな開発ニーズに対応するために、JavaScriptベースの独自の統合フレームワークで開発されました。nexacro platformで開発されたアプリケーションは、別の追加開発なしに、さまざまなオペレーティングシステム、ブラウザ、デバイスで同じ機能を実装することができます。

nexacro platform 14から提供する、ウィジウィグ(WYSIWYG)ベースの開発ツールであるnexacro studioは、RTE(Real Time Enterprise)実現のための開発生産性を最大化するために、HTML5、ランタイムバージョンに関係なく開発することができ、開発者の利便性を最大化すると同時に、さまざまなデザイン効果を簡単に与えることができるようにサポートします。

nexacro platform 14は、システムのUIにアクセスする状況によってHTML5とランタイムバージョンに区分され、同じ開発プロセスと設定ファイルを使用します。HTML5バージョンは、別の追加インストールせずに配布の問題から非常に自由なのが特徴であり、ランタイムバージョンは仮想エンジンのインストールによってデスクトップ、スマートフォン、タブレットなど、さまざまな環境で相対的に高い性能と信頼を確保することができます。ランタイムバージョンは、オペレーティングシステムによって、最適化された仮想エンジンを提供します。

1.2.1 主な特徴

- 強力な開発ツール

  既存の4GL開発ツールのように、開発に必要な多様で便利な機能をnexacro platform専用開発ツールであるnexacro studioで提供します。ウィジウィグ(WYSIWYG)方式の画面開発と共に知能的なエディタを提供し、開発者に便利な機能をサポートします。

- 標準の準拠

  nexacro platformで使用するスクリプト言語は、ECMA、CSS、XML、DOMなどの国際標準に準拠した汎用技術を適用することにより、他のシステムまたはソリューションとの柔軟なインターフェイスをサポートします。

  標準技術を使用して新しい言語を学ばなければならないという負担を減らすことができ、nexacro platformの開発環境を迅速に習得して簡単に開発および保守が可能です。

- 統合されたデバイスのイベント処理

  PCでのマウスイベント処理とスマートデバイスでのタッチイベント処理を標準化して、一つのコードでさまざまなデバイスをサポートするイベント処理を実装することができます。

- 高速のデータ処理

  nexacro platformは、サーバーとクライアント間のデータ処理時の画面と情報を除いたデータのみ送受信することにより、サーバーの負荷を削減し、ネットワークの効率を高め、高速の通信速度を実現することができます。

  非同期通信技術を利用したデータ通信をサポートし、業務によって最適な実行環境を作ることができます。

- 企業に特化した豊富なコンポーネント

  アプリケーションですぐに使用できるさまざまな企業用コンポーネントを提供してアプリケーションに必要なコンポーネントの開発期間を最小限に抑えることで、開発期間を短縮し、変化するビジネス環境に速やかに対応できます。特に最も多く使われるGridは、企業で主に使う機能を内蔵して提供しています。

  nexacro studioでは、コーディングせずに、ドラッグ&ドロップ操作だけで各コンポーネントとのデータバインディングすることができ、クリック操作だけで新しいイベントを作成することができます。

- 視覚効果

  さまざまな視覚効果を処理したり、スタイルやテーマを編集する作業をnexacro studio内で簡単に実装することができ、開発者とデザイナーの両方に柔軟なコラボレーションプロセスをサポートします。

- 企業専用ブラウザ(ランタイムバージョン)

  Real Time Enterprise(RTE)の実現のために企業で使用するアプリケーションで、機能や性能は重要な要素です。ウェブブラウザの性能は持続的に向上してきているが、企業が必要とする安定した性能への期待を満たすことができない限界を持っています。それらの限界から脱皮するために、ランタイムバージョンで提供する企業専用ブラウザ機能を活用することができます。

1.2.2 適用対象システム

- 既存のシステムのUI高度化によるTCOの削減とROIの向上
  - メインフレーム環境など既存のシステムをウェブに移行
  - 使用が複雑なSAP UIの改善/高度化
  - 既存のシステムの老朽化でUI構成が複雑になるにつれ使用度が低下されたシステムを改編
  - コールセンターシステムのようにUIの改善/再構築時に顧客満足度の向上、人当たり処理速度向上によるコスト削減
  - ERP、CRM、PLM、SCMなどに対するシステムアクセシビリティを向上
- クライアント/サーバーシステムをインターネットベースのシステムに移行
  - クライアント/サーバーシステムのインストール、配布、更新時の問題を根本的に解決

    - 純粋なウェブでサポートされていないオフライン駆動とLocal DBMSやLocal Fileとの連動業務
- スマートフォンやタブレットなど、さまざまな機器のサポートが必要なシステム構築
    - PCとスマートフォン、タブレットまで共にサポート
    - BYOD(Bring Your Own Device)環境サポートシステムの構築
- 数値、統計データ中心の対顧客ウェブサービスやアプリケーションの開発
- 代表取締役、役員または意思決定者を対象に、直感的で流麗なEISや状況板などのシステム構築
- HTML5を収容した次世代のシステム構築

第2章

nexacro platformの概要

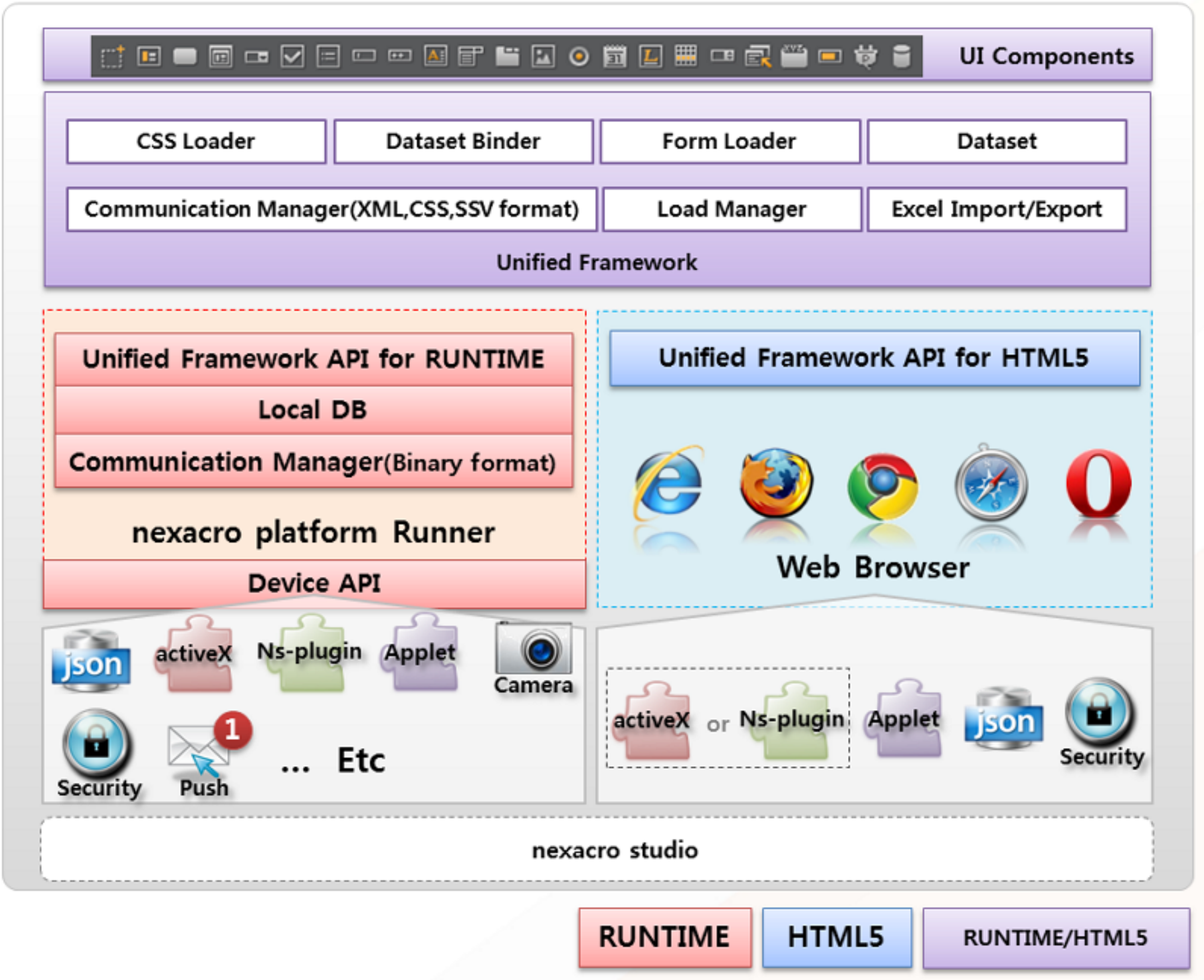
nexacro platformは、さまざまなデバイスやオペレーティング環境をサポートするために、統合フレームワーク(Unified Framework)とJavaScriptベースのコンポーネントを提供します。一つのコードでどのような環境でも動作するアプリケーションを作成することができ、各動作環境によって最適化された実行環境を提供します。

2.1　nexacro platform統合フレームワーク

nexacro platform統合フレームワークは、さまざまな実行環境で最適なパフォーマンスを提供し、新しいオペレーティングシステムが登場してもすぐに対応できる構造に作られました。

バージョンによって、パッケージや配布方法は異なることもありますが、内部的に動作するアプリケーションは同じなので、どんな環境でも同じ動作を保証することができます。アプリケーションで使用するすべてのコンポーネントは、JavaScriptベースに作られ、Webブラウザとnexacro platformランナー(Runner)で同じように動作します。ただし、Webブラウザは、ユーザーが使用しているバージョンによって性能が異なる場合があります。

統合フレームワークは、他の要素と一緒にnexacro platformエンジンの役割を果たします。主な要素は次の通りです。



構成要素	説明
統合フレームワーク (Unified Framework)	プロジェクトとアプリケーションを構成する基本的なフレームワークを提供します。データ通信(XML、CSVなど)、画面間の連携のような作業を処理し、設定されたテーマに基づいて全体のスタイルを処理する機能が含まれています。
UIコンポーネント (UI Components)	画面上に表示されるUIコンポーネントとDatasetのようなオブジェクトを提供します。内部的にはUIコンポーネントは、基本コンポーネントを組み合わせて作成されます。 - UIコンポーネント：Button、Calendar、Checkbox、Gridなど - デフォルトコンポーネント：EditBase, FormBase, FrameBase, ScrollBarなど - オブジェクト：Dataset
統合フレームワークAPI	アプリケーションは、統合フレームワークで構成され、ランタイムとHTML5バージョンが同じ方式で動作し、開発者は、どのバージョンを使っても同じ方法で開発します。内部要素であるエレメント(Element)は、それぞれの環境に最適化された方式で開発されました。ランタイムバージョンはCをベースに作成され、HTML5バージョンはJavaスクリプトをベースに作成されます。但し、開発者がカスタムコンポーネントを作成したり、プロジェクトに必要な共通の作業を行うときは、JavaScriptベースとして、一つのコードのみを使用します。残りは、統合フレームワーク内で、使用環境によっ

構成要素	説明
	て必要な要素を適用して処理します。
Device API	サードパーティが開発したモジュール(Extension DLL)との連携時に使用する機能です。外部機器連動などの処理をサポートします。
Device API (Camera...)	モバイル機器の使用時にカメラやダイヤルなどの機器自体の機能との連携時に使用する機能です。機械によってサポートされる機能が制限されることがあります。 - Webブラウザによって、一部の機能を提供しますが、まだ実験段階のため、HTML5バージョンではサポートしておりません。
Local Database	アプリケーション内でデータを保存して処理することができる機能をサポートします。SQLiteをサポートし、デスクトップだけでなく、モバイル機器でも使用できます。- Webブラウザでも Indexed Database(http://www.w3.org/TR/IndexedDB/)をサポートしますが、まだ実験段階のため、HTML5バージョンではサポートしておりません。
バイナリデータ通信	データ通信時にXML、CSV、SSVなどの形式は、ランタイム、HTML5バージョンの共に基本的にサポートし、ランタイムバージョンではバイナリ通信をさらにサポートします。 - Webブラウザでもバイナリデータ通信処理が可能ですが、まだ実験段階のためHTML5のバージョンではサポートしておりません。

2.2 nexacro platform構成要素

nexacro platformは、4つの要素で区切られます。プログラミング言語と開発環境は、製品のバージョンに関係なく同じであり、実行環境と開発環境は、バージョンによって異なる場合があります。

2.2.1 プログラミング言語

nexacro platformは、他のプログラミング言語とは異なり、ユーザーに表示される画面を定義する部分とビジネスロジックを処理するスクリプトで区切られます。また、画面に好きなデザインを適用するために、スタイルとテーマを適用することができる機能を提供します。

画面を配置する部分は、XMLベースで各コンポーネントのプロパティとバインド、イベントなどの情報を管理します。また、様々な実行環境をサポートできるようにMLM(Multi Layout Manager)機能をサポートし、関連するプロパティを管理します。また、アプリケーションの実行に必要な環境情報を管理する別のファイル構造を提供します。

以下は、nexacro platformでのアプリケーション開発時にデフォルトで作成されるファイルに関する説明です。

区分	ファイル名(拡張子)	用途
nexacro platform Project	*.xprj	• プロジェクト情報 • TypeDefinition • グローバル変数 • ADL (アプリケーション)
nexacro platform Application Definition	*.xadl	• アプリケーション実行環境 • TypeDefinition • グローバル変数 • テーマ情報 • フレームプロパティ • スクリーン情報(ScreenInfo)
nexacro platform Form Definition	*.xfdl	• 画面レイアウト • 画面フォームプロパティ • コンポーネントプロパティ • 追加レイアウト • スクリプト
TypeDefinition	default_typedef.xml	• モジュール • コンポーネント

区分	ファイル名(拡張子)	用途
		• サービス • アップデート
GlobalVariable	globalvars.xml	• グロバール変数
Theme	*.xtheme	• スタイルシート (フレーム、フォーム、コンポーネントなど) • イメージ

生成されたアプリケーションは、ビルドプロセスを経て、JavaScriptベースのコードに変換されます。実際の実行環境では、変換されたJavaScriptコードを実行するようになります。

2.2.2 開発環境

nexacro platformは、ウィジウィグ(WYSIWYG)ベースの開発ツールであるnexacro studioを提供します。nexacro studio内で実行環境に関係なく、アプリケーションを開発することができ、生成されたコードは、nexacro platformのプログラミング言語で保存されます。

nexacro studioは、Microsoft Windowsオペレーティングシステムのみサポートしますが、開発されたアプリケーションは、どんなオペレーティングシステムやどんなデバイスでも関係なく、最適化された使用環境で配布することができます。

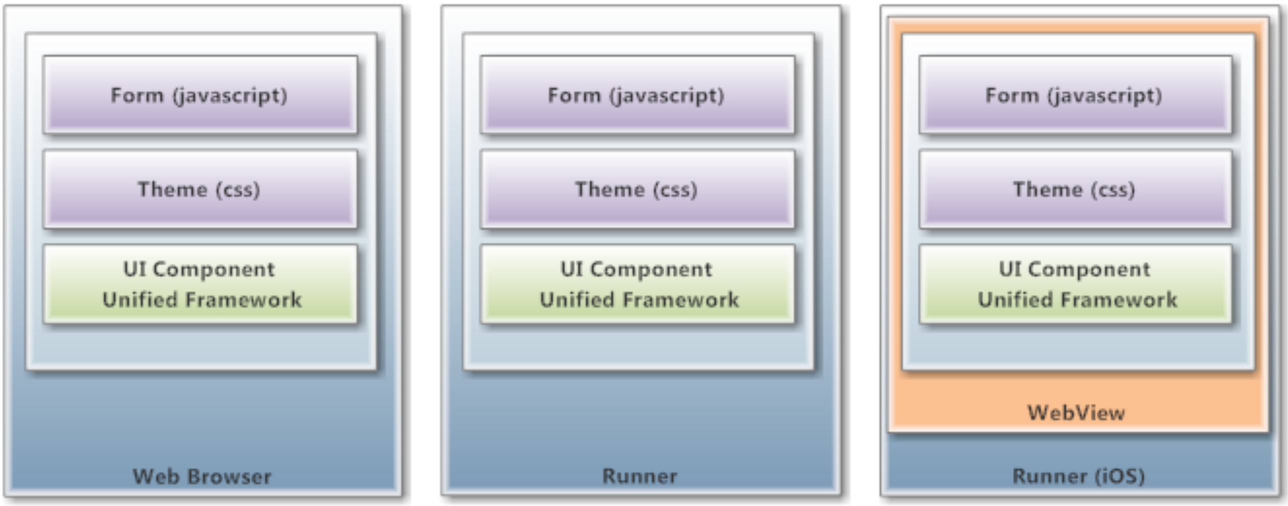
> 配布環境によって、nexacro studio以外の開発ツールで追加的な作業を必要とすることができ、別々の認証を受けなければなりません。たとえば、iOSをサポートするアプリを開発する場合には、配布時にXcode環境が必要であり、アンドロイドOSをサポートするアプリを開発する場合には、配布時にEclipse環境が必要となります。

2.2.3 実行環境

nexacro platformは、統合フレームワーク(Unified Framework)を元に、アプリケーションが実行され、各実行環境によって最適化された構造を提供します。

アプリケーションを実行する主体によってランタイムとHTML5の2つのバージョンが提供されます。ランタイムバージョンは、各オペレーティング·システムによって最適化された実行環境を提供します。新しいオペレーティングシステムが発売されても、必要なAPIを追加してサポートすることができます。 HTML5バージョンは、ユーザーのWebブラウザ基に動作します。

nexacro platformが実行される環境を簡単に表現すると次の通りです。



ランタイムバージョンは、独自に開発されたレンダリングエンジンを含むnexacro platformランナー(Runner)を提供します。そのため、一般的なWebブラウザよりも高速安定した性能を保証することができます。

ランタイムバージョンでも、オペレーティングシステム上の制約がある場合があります。たとえばiOSで実行されるアプリはUIWebViewというオブジェクトのみ使用しなければなりません。同じフレームワークを使用しますが、オペレーティングシステムメーカーのポリシー上で、UIWebViewオブジェクトを使ってもう一度包み込む形式をとります。

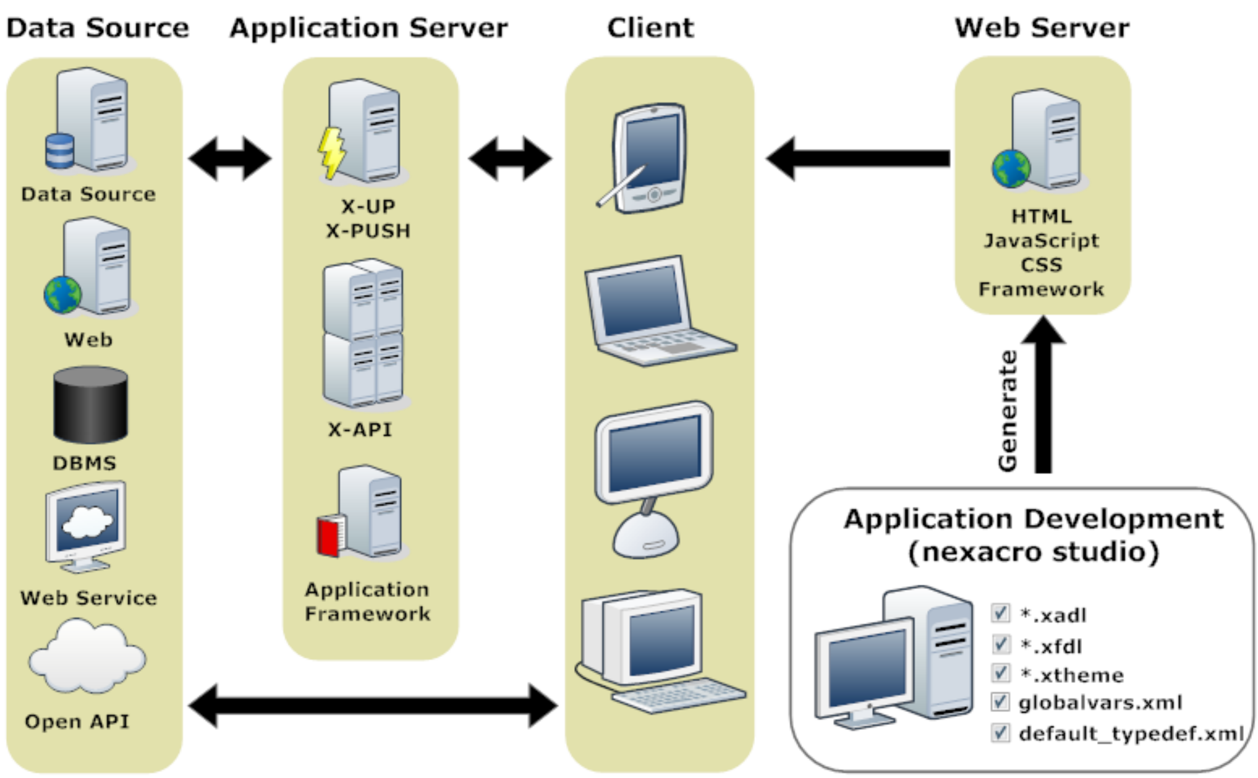
2.2.4 配布環境

配布とは、アプリケーションが実行されるために必要なリソースをクライアントにインストールするための一連の作業を意味します。使用環境によって、nexacro platformで開発されたアプリケーションと必要なモジュールをダウンロードしてクライアントにインストールするようになります。

nexacro platformのアプリケーションは、ユーザーが使用するクライアントで動作します。ただし、データ処理などの作業のために、アプリケーション·サーバー(Application Server)を必要とすることがあります。nexacro platformは、データ処理のためのX-APIモジュールを共に提供します。また、必要によってデータをリアルタイムで処理する必要がある場合、X-PUSHと同じような付加的な技術を使用することができます。

基本的な配布作業は、HTTPプロトコルを使用します。ただし、インターネット接続をサポートしていない環境では、アプリケーションの実行に必要なリソースを別々のメディアに配布して使用することができます。

各配布環境によって、追加の作業が必要になることがあります。配布と関連した詳細内容は管理者ガイド(Administrator Guide)で提供される各バージョンごとの詳細を参照してください。



X-PUSHは、ランタイムバージョンのみサポートします。
ランタイムバージョンでは、インストールモジュールが実行環境によって配布されます。

テーマファイルは、組み込みテーマを使用する場合には、nexacro studioで別々に作成せずにデフォルトテーマを変換する処理だけ行います。

2.3 システム要件

nexacro platformアプリケーションが動作するための環境は、次のとおりです。

- Runtime

項目	仕様	Windows	Android	iOS
CPU	最小	Intel® Pentium® 4 1.4GHz	ARM Cortex-A8 600 MHz	ARM Cortex-A8 600 MHz
	推奨	Intel® Core™ i3-330M 2.13GHz	ARM Cortex-A8 1GHz以上	ARM Cortex-A8 1GHz以上
ディスプレイ	最小	-	HVGA	HVGA
	推奨	-	HVGA以上	HVGA以上
メモリ	最小	1GB	256M	256M
	推奨	4GB DDR3	512M	512M
HDD(ROM)	最小	8GB	4M	4M
	推奨	30GB - 5400rpm	10M	10M
プラットフォーム		Windows XP SP3以上	Android 2.3以上	iOS 9以上

- HTML5

Webブラウザ	最小仕様	推奨仕様	最高仕様
Google Chrome	10.0.648.127	22.0.1229.79	最新バージョン
Firefox	8.0.1	14.0.1	最新バージョン
Internet Explorer	8.0	10.0.9200.16521	11.0
Edge	25.10586.63.0	25.10586.63.0	最新バージョン
Safari	4	6.0.2	最新バージョン

モバイルWebブラウザは、下記のブラウザのみをサポートします。
- Androidデフォルトブラウザ
- Android Google Chromeブラウザ
- iOSデフォルトブラウザ

モバイルWebブラウザ	最小仕様	推奨仕様	最高仕様
Androidデフォルトブラウザ	Android 2.3 デフォルトブラウザ	Android 6.0 デフォルトブラウザ	Android 7.0 デフォルトブラウザ
Android Google Chromeブラウザ	Android Chrome (Android 4.0)	Android Chrome (Android 4.4)	Android Chrome 最新バージョン
iOSデフォルトブラウザ	iOS 9 デフォルトブラウザ		iOS 10.1 デフォルトブラウザ

第3章

nexacro platformの配布

ユーザーが、nexacro platformで開発されたアプリケーションを使用するには、使用環境によって適切な実行環境を作ってくれて、必要なファイルをダウンロードする必要があります。このような過程を’配布’と言います。配布されたファイルはキャッシュポリシーによって、内部で管理され、必要なときに新しいファイルに置き換えられます。

nexacro platform HTML5バージョンの場合には、既存に使用していたWebブラウザ環境でアプリケーションが実行されるため、Webサーバーから必要なファイルをダウンロードして、すぐに実行します。ただし、ランタイムのバージョンは、各実行環境によって、追加インストール作業を必要とします。ランタイムバージョン配布についた詳細な内容は、各実行環境による内容を参考してください。

3.1　概要

nexacro platformアプリケーションを実行するために必要なファイルを先に説明しましょう。実行環境によって配布されるファイルは、次のとおりです。

分類	説明	windows	iOS/andriod	HTML5
Runtime Engine	スクリプト、メモリ、レンダリング処理	O	△ (iOS: WebView)	X (Webブラウザ)
plugin	ブラウザごとプラグイン	O	X	X
Fremework	統合フレームワーク (JavaScript)	O (XFrameworkLib.dll)	O	O
Component	コンポーネントライブラリ (JavaScript)	O	O	O
Application	アプリケーションコード (JavaScript)	O	O	O

nexacro platformランタイムとHTML5バージョンの最も大きな違いは、ランタイムエンジンです。HTML5バージョンは、Webブラウザで必要なフレームワーク、コンポーネント、アプリケーションをすぐに使用するのに対して、ランタイムバージョンは、各実行環境に適したランタイムエンジンを個別に配布する必要があります。

ランタイムエンジンを追加で配布する作業がユーザーの立場からも不便なことがありますが、システム環境によって、既存に使ったシステムとの連携が必要な場合や、Webブラウザで提供していない拡張された機能を使用するためには、ランタイムバージョンを選択することができます。

> nexacro platformで提供する基本的な機能は、バージョンに関係なく同じですが、デバイスAPIや外部機器連動などの拡張機能は、ランタイムバージョンでのみ提供されています。

3.2 配布ファイル

各バージョンごとに配布されるファイルを詳細に調べてみると以下のとおりです。

ファイルの位置またはサーバーの位置を説明するAliasは、Aliasパス を参考してください。

以下に説明されたファイル名と提供されるライブラリは、アップデートによって変更される場合があります。

3.2.1 Runtime for windows

nexacro platformランタイムエンジンの配布後に実行方法によってアプリケーションを呼び出します。コンテンツはHTML5バージョンと同じですが、フレームワークのJavaScriptファイルをサーバーから直接受けずにdllファイルとして提供されたライブラリに含ませて配布します。

分類	ファイル	説明	ファイル位置
Runtime Engine	V8.dll	V8 Script Engine Library	%nexacro%
	XMemLib.dll	Memory Management Library	%nexacro%
	XBasicLib.dll	Base Libaray	%nexacro%
	nexacrolib.dll	Platform & Render Engine Library	%nexacro%
	nexacro.exe	nexacro platform Executor	%nexacro%
ActiveX DLL	nexacrolauncherax14.dll	nexacro platform Launcher	%SYSTEM32%
	nexacroax14.dll	BrowserごとのPlug-in	%nexacro%
Plugins DLL	npnexacrolauncher14.dll	nexacro platform Launcher	%SYSTEM32%
	npnexacro14.dll	BrowserごとのPlug-in	%nexacro%
Framework DLL	XFrameworkLib.dll	Framework API Library	%nexacro%
	XMobileApi.dll	Device API Library	%nexacro%
Extend DLL Files	protocol adaptor dll	Network Protocol Adaptor	%UPDATE%
	External DLL	External DLLs	%UPDATE%
Runtime Cache	cache.db	Runtime Cache DB	%CACHE%

分類	ファイル	説明	ファイル位置
	cachedfiles	Runtime Cached Data Files	%CACHE%\key_adl
Runtime Config Files	nexacro.xml	Runtime 実行環境ファイル & UserProfile	%USERAPP%
Log Files	nexacro_xxxxx.log	ログファイル	%USERAPP%\Log
Launcher Download Files	splash.png	Splash Image File	%USERAPP%
	loadingimage.png	WaitCursor Image File	%USERAPP%
	nexacro.ico	Application Icon File	%USERAPP%
	globalvars.dat	Global Variable Listファイル	%USERAPP%

> ファイル名は、製品バージョンや国別のライセンスポリシーに基づいて異なる場合があります。

3.2.2 HTML5

必要なすべてのファイルをWebブラウザから直接ロードする方法で実行されます。使用しているWebブラウザによって、性能の違いがある場合があります。

分類	ファイル	説明	サーバー位置
Execute HTML	index.html		%WEBDEPLOY PROJECT%
Framework JS	BasicObjs.js		%WEBDEPLOY FRAMEWORK%
	SystemBase.js	System Utility	%WEBDEPLOY FRAMEWORK%
	Platform.js	Platform Objects	%WEBDEPLOY FRAMEWORK%
	CssObj.js	Style Object	%WEBDEPLOY FRAMEWORK%
	ErrorDefine.js	Error Information	%WEBDEPLOY FRAMEWORK%
Component JS	CompBase.json	Component Base Module	%WEBDEPLOY COMPONENT%
	ComComp.json	Common Component Module	%WEBDEPLOY COMPONENT%
	Grid.json	Grid Component Module	%WEBDEPLOY COMPONENT%
	DeviceAPI.json	DeviceAPI Module	%WEBDEPLOY COMPONENT%
Resource Files	waitimage.gif	WaitCursor Image File	%WEBDEPLOY RESOURCE%
Theme File	defaulttheme.js	Deployされたテーマソース	%WEBDEPLOY THEME%
Application Files		Deployされたアプリケーションソース	%WEBDEPLOY PROJECT%

3.2.3 Runtime for iOS/Android

iOS/ Androidのアプリは、各OSによって、別のパッケージングプロセスが必要になります。nexacro studioで作成されたアプリケーションのソースをArchiveファイルで作成し、各OSに合わせて配布ファイルを生成します。

分類	ファイル	ファイル説明	Packing
Android App Files	nexacro14.jar	Android Library Jar	Android App
	libnexacro14.so	Android Library SO	
	start_android.json	Android Archive Information	
	strings.xml	Android Locale String Information	
iOS App Files	nexacro14.framework	iOS Library Framework	iOS App

分類	ファイル	ファイル説明	Packing
	start_ios.json	iOS Archive Information	
	localizable.strings	iOS Locale String Information	
Execute HTML	Run.html		Run Archive
Framework JS			Engine Archive
Component JS			Engine Archive
Resource Files	loadingimage.png		Engine Archive
Theme File	defaulttheme.js		Theme Archive
Application Files			Application Archive

第II部

Runtime for windows

第1章

nexacro platformの配布概要

nexacro platformで開発されたサービスを、Windows PCユーザーに提供するには、nexacro platformの実行環境をユーザーのPCにインストールして開発されたアプリケーションをダウンロードする必要があります。これらのインストール手順を配布と言います。配布はnexacro platformエンジンによって自動的に行われ、HTTPプロトコル(Protocol)を使用したり、オフラインで手作業で行うことができます。

1.1　配布フロー

nexacro platformランタイムでの配布は、エンジンを配布して初めて起動するHTMLファイルを呼び出す部分まで管理します。実際のユーザーが見るnexacro platformアプリケーションは、HTMLファイルとJavaScriptファイルに対する処理順序に従います。

1. nexacro platformエンジンの配布手順

   配布の手順を全般的に管理するnexacro platformエンジンをインストールします。

   Webブラウザの特定のページを通じてダウンロードしてインストールするか、exeファイル形式のインストールファイルを使用してインストールします。Webブラウザでダウンロードしてインストールする場合には、当社が提供するCABファイルを利用します。

2. 統合フレームワークとコンポーネントライブラリの配布手順

   フレームワークは、エンジンと一緒に配布されるフレームワークDLLファイルにリソースファイルとして配布されます。

3. アプリケーションの配布手順

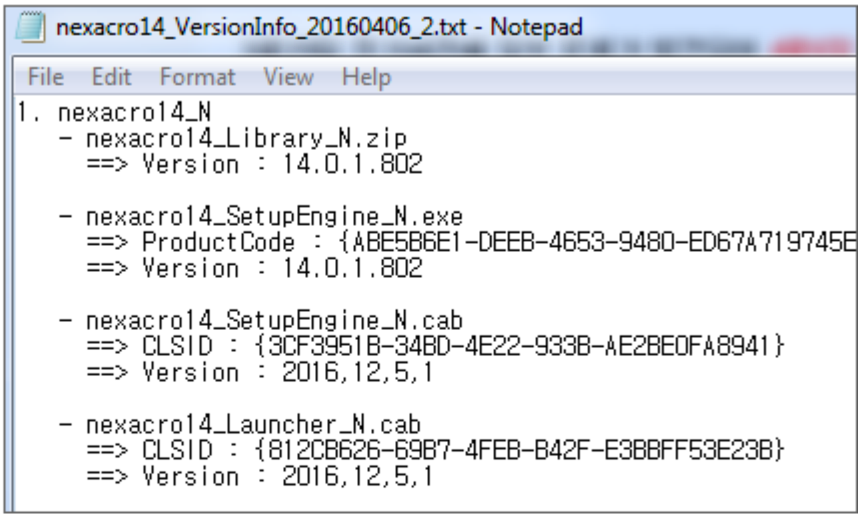
   開発者が開発したアプリケーションを配布します。アプリケーションの配布時コンポーネントモジュールも一緒に配布されます。

   アプリケーションの配布は向上されたキャッシュ管理(Intelligent Cache)の手法を使って効率的にネットワークを使用します。

1.2 配布の段階

ほとんどの配布はnexacro platformエンジンにより実行されます。では、nexacro platformエンジンはどのようにして使用者の端末機に設置されるのでしょうか。nexacro platformエンジンは、主にウェブブラウザによりCABファイルで設置されます。ウェブブラウザを使わない場合は、実行形式インストーラを別途に提供します。

> 以下の説明に含まれたcabファイル、classid、CodeBase項目は、製品のバージョンや国別ライセンスポリシーに基づいて異なる場合があります。製品と一緒に配布されているnexacro14_VersionInfo_XXX.txtファイルを参照してください。


nexacro14_VersionInfo_20160406_2.txt - Notepad
File Edit Format View Help

```
1. nexacro14_N
   - nexacro14_Library_N.zip
     ==> Version : 14.0.1.802

   - nexacro14_SetupEngine_N.exe
     ==> ProductCode : {ABE5B6E1-DEEB-4653-9480-ED67A719745E
     ==> Version : 14.0.1.802

   - nexacro14_SetupEngine_N.cab
     ==> CLSID : {3CF3951B-34BD-4E22-933B-AE2BE0FA8941}
     ==> Version : 2016,12,5,1

   - nexacro14_Launcher_N.cab
     ==> CLSID : {812CB626-69B7-4FEB-B42F-E3BBFF53E23B}
     ==> Version : 2016,12,5,1
```

1.2.1 1段階：nexacro platformエンジンの設置

ウェブブラウザはCABファイルを利用してnexacro platformエンジンを設置します。CABファイル(nexacro14_SetupEngine_N.cab)のインストールのためにウェブブラウザが最初にアクセスするHTMLは、以下のようなコードが挿入される必要があります。

```
<HTML>
<HEAD>
<TITLE> Plugin sample page </TITLE>
<BODY>
<object id="NexacroAXCtrl"
```

```
    classid="clsid:3CF3951B-34BD-4E22-933B-AE2BE0FA8941"
    CodeBase="./nexacro14_SetupEngine_N.cab#VERSION=2014,7,11,2">
</object>
</BODY>
</HTML>
```

ここでのバージョン番号は設置するモジュールのバージョンに依存するため、その値が異なる場合があります。詳しくは、製品と一緒に配布されるVersionInfoファイルを参照してください。

1.2.2　2段階：nexacro platform応用プログラムの実行環境の設定

nexacro platformエンジンは、最初の応用プログラムの配布及び環境設定のためにADLファイルをダウンロード及び解析します。Aと言う応用プログラムを実行するには、A応用プログラムの環境情報を格納しているADL(拡張子：xadl)ファイルがなければなりません。

ADLはApplication Definition Languageの略で、アプリケーションの構成に必要な要素を定義する文書としてxadl拡張子で保存され、配布時には、JavaScriptファイルにType definitionとGlobalVariable情報が含まれています。

そのほかに、基本画面配置、コンポーネント及びオブジェクト構成、テーマ構成などの要素を定義します。

1.2.3　3段階：応用プログラムの実行

nexacro platformエンジンは、ADLの内容の通りに応用プログラムの設定を終えてからUI画面をダウンロードして表示します。

使用者のUI情報はFDLファイルに格納されており、拡張子はxfdlです。

FDLはForm Definition Languageの略で、ユーザーインターフェイスの情報と、画面内で処理されるロジック情報を含んでいます。 FDLファイルは、xfdl拡張子として保存され、配布時には、JavaScriptファイルに変換されて提供されます。

1.3　設置用ファイル

当社は、nexacro platformの配布のために様々な設置用のファイルを提供します。

1.3.1　セットアップファイル

以下の表は、nexacro platformのセットアップファイルに含まれる主要モジュールです。

セットアップファイル名	
nexacro14_SetupDeveloper_N.exe	• nexacro platform Engine (nexacro.exe + library dll) • nexacro14lib (JavaScript) • nexacro studio
nexacro14_SetupEngine_N.exe nexacro14_SetupEngine_N.cab	• nexacro platform Engine • nexacroax14.dll • nexacrolauncherax14.dll
nexacro14_Launcher_N.cab	• nexacrolauncherax14.dll

1.3.2　ライセンスファイル

以下の表は、ライセンスファイルの種類です。

ライセンスファイル名	
nexacro14_server_license.xml	XML形式のファイルで提供します。 X-API jarファイルと同一位置またはclassesフォルダに位置します。 X-APIとHTML5の実行時にチェックします。
nexacro14_client_license.xml	XML形式のファイルで提供します。 ウェブでアクセス可能なパスに位置します。 ADLのlicenseurlに該当ファイルへのパスとファイル名を設定します。 ランタイムとウィジェットの実行時にチェックします。
Tool License	プロダクトキー形式の文書で提供します。 nexacro studioのHelp – About nexacro studioで登録します。

1.3.3　主要モジュールの機能

以下の表は、配布される主要モジュールの機能です。

モジュール名	主要機能
nexacrolauncherax	Iconのイメージをダウンロードします。 ショートカットを生成します。 nexacro platformエンジンを起動させます。
nexacroax	nexacro platformエンジンをウェブブラウザ上で駆動させます。

1.4 nexacro platformの配布の種類

nexacro platformの配布は、nexacro platformランタイムがどのような形式で実行されるのかによって、配布方法が異なります。

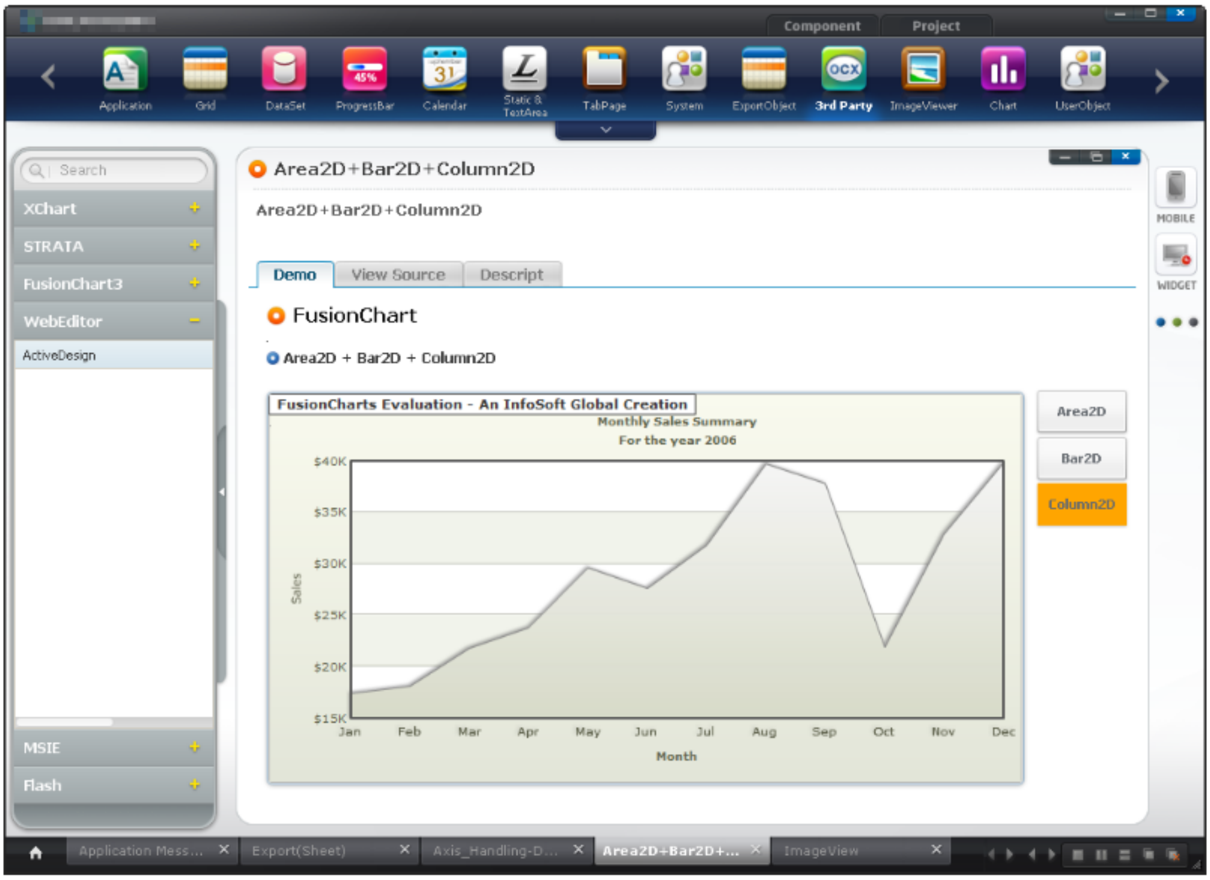
1.4.1 nexacro platformランタイムの実行形態

nexacro platformランタイムのコア部分であるエンジンが、どのような形態で実行されるのかにより2種類に分けることができます。“Executionバージョンの実行”と“ウェブブラウザプラグインバージョンの実行”の2種類です。

また、その実行形態により配布方法も異なります。

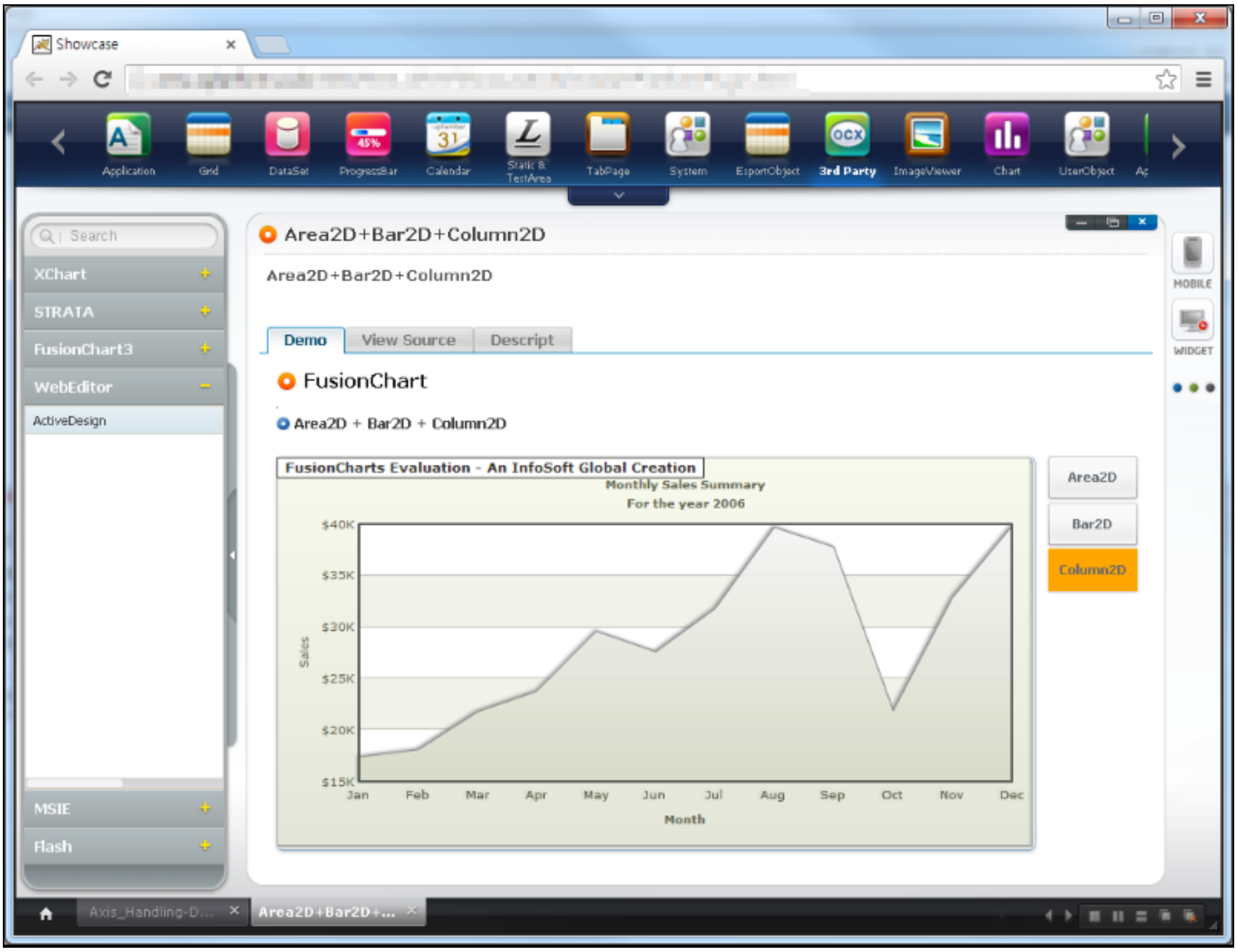
Executionバージョンの実行

“Executionバージョンの実行”は、nexacro platformランタイムを独立したプロセス(Runner)で実行することを意味します。Launcherによる実行とexeファイルの実行の2種類の方法があります。



ウェブブラウザプラグインバージョンの実行

“ウェブブラウザプラグインバージョンの実行”は、ウェブブラウザに組み込まれた状態でnexacro platformランタイムを実行することを意味します。



> Web Browser Pluginで実行される形態は、独立して実行されますが、ユーザー環境(Webブラウザ)の影響を受けるため、予想外の問題が発生することがあります。
> Webブラウザ上での実行をご希望する場合は、nexacro platform HTML5バージョンを検討してください。

1.4.2　基本配布の種類

nexacro platformの配布は、nexacro platformの実行形態により3種類に分けられます。

“Plugin配布”、“Launcher配布”、“exe配布”の3種類に分けられ、それぞれの内容を以下の表により比較しました。

<table>
<tr><th>種類
項目</th><th>ウェブブラウザ
Plugin配布</th><th>Launcher配布</th><th>exe配布</th></tr>
<tr><td>Running
Window</td><td>Web Browser</td><td>nexacro platform Runtime</td><td>nexacro platform Runtime</td></tr>
<tr><td rowspan="2">設置ファイル</td><td rowspan="2">nexacro14_SetupEngine_N.
cab</td><td>nexacro14_SetupEngine_N.
cab</td><td rowspan="2">nexacro14_SetupEngine_N.
exe</td></tr>
<tr><td>nexacro14_SetupEngine_N.
exe
nexacro14_Launcher_N.cab</td></tr>
<tr><td>エンジンの
実行</td><td>nexacro platform ActiveXの
実行</td><td>nexacrolauncheraxの実行</td><td>nexacro.exe の実行</td></tr>
<tr><td>ショートカットの使
用</td><td>なし</td><td>ショートカットの生成</td><td>ショートカットの実行</td></tr>
</table>

ショートカットを使用するには、“Launcher配布”方式と“exe配布”方式を複合的に使用しなければなりません。Launcherを利用してショートカットを生成し、exe実行方式によりショートカットをクリックして実行します。

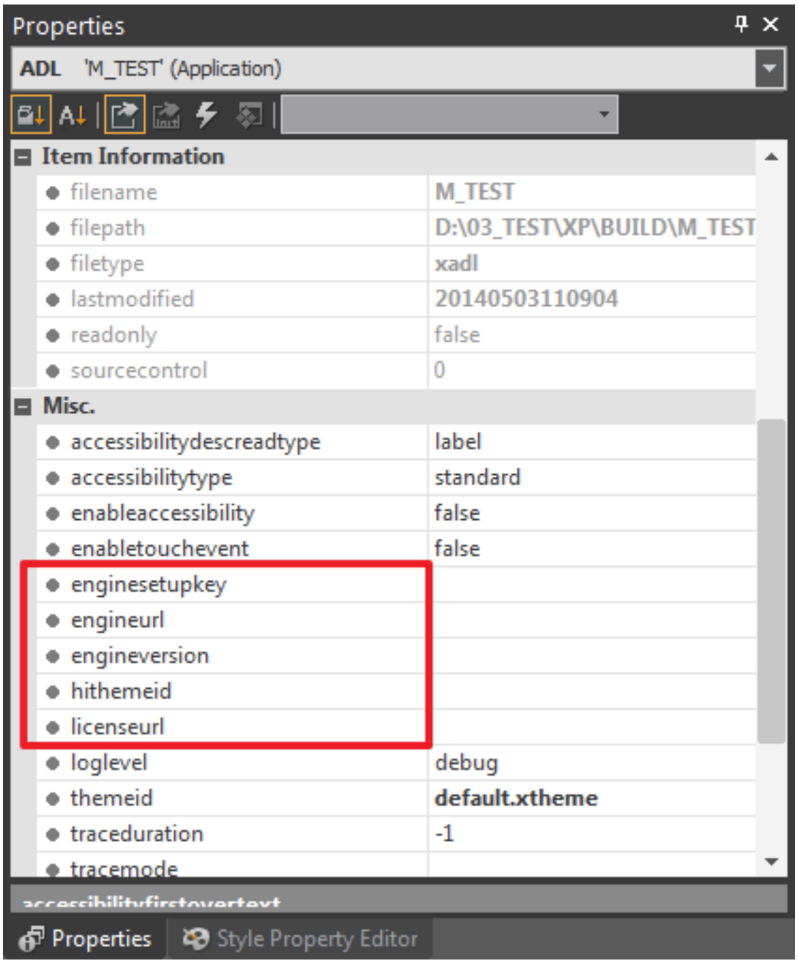
1.5 nexacro platformエンジンのバージョン管理

ほとんどの配布は、nexacro platformエンジンが管理します。nexacro platformエンジンは主にウェブブラウザによって配布されると説明しました。では、nexacro platformエンジン自体がアップグレードされた時にはどのように配布するのでしょうか。nexacro platformエンジンが自らアップグレードします。

1.5.1 nexacro platformエンジンのアップグレード情報の登録

nexacro platformエンジンがADLファイルをロードし応用プログラムの実行環境を設定する段階で、アップグレードするか否かを決定して実行します。即ち、エンジンに対するアップグレード情報はADLファイルに格納されています。

以下の画面は、nexacro studioのADLプロパティでアップグレード情報を登録する画面です。



各項目の詳細説明は以下の通りです。

プロパティ名	説明
enginesetupkey	セットアップファイルのProductキーです。 nexacro platformエンジンを使用する時には何の値も入力しません。しかし、開発者が別途のセットアップファイルを作成する時には、該当するセットアップファイルのProductキーを入力します。
engineurl	セットアップファイルが存在するサーバーのURLパスです。
engineversion	セットアップファイルのバージョンです。
licenseurl	ライセンスファイルのパスを設定します。

nexacro platformエンジンのインストール時に使用されたHTMLファイルに使用された値は、ADLプロパティ値と、下記のような関係を持っています。

```
<HTML>
<HEAD>
<TITLE>Plugin sample page </TITLE>
<BODY>
     <object  id="NexacroAXCtrl"
          classid="clsid:3CF3951B-34BD-4E22-933B-AE2BE0FA8941"
          CodeBase="./nexacro14_SetupEngine_N.cab# ← engineurl
          VERSION="2014,7,11,2"> ← engineversion
     </object>
</BODY>
</HTML>
```

1.5.2　nexacro platformエンジンのアップグレードの実行

nexacro studioでADLプロパティ値を入力してからADLファイルを配布サーバーに格納すると、nexacro platformエンジンがADLファイル内のengineurlとengineversion値を現在設置されているnexacro platformエンジンと比べてアップグレードします。

既に設置されているエンジンのバージョンは、使用者のコンピュータの“コントロールパネル/プログラムの追加/削除”で確認することができます。

nexacro platformエンジンをアップグレードする場合には、nexacro platformエンジンが再起動されます。

第2章

基本配布方法

nexacro platformの基本配布には“Plugin配布”、“Launcher配布”、“exe配布”の3種類があります。この3種類の基本配布を組み合わせて、より高度な配布を行うことができます。

2.1　Plugin配布

Plugin配布はnexacro platformランタイムがWebブラウザに埋め込まれた状態でアプリケーションを実行するときに使用します。ユーザーがWebブラウザを脱していない状態で、他の作業と一緒に仕事を進めることができ、お気に入りに追加して再度アプリケーションにアクセスすることができます。

サービス管理者は、配布時、下記のようなディレクトリ構造で管理することができます。

Path	File / Folder	説明
./	nexacroAX.html	インターネットエクスプローラの使用時の配布ページ
./	nexacroPlugin.html	Chrome、Firefox、Safariなどの使用時の配布ページ
./	nexacro14_SetupEngine_N.cab	インターネットエクスプローラの使用時のnexacro platformランタイムエンジン（Active X）をインストール
./	nexacro14_SetupEngine_N.exe	Chrome、Firefox、Safariなどの使用時にnexacro platformランタイムエンジン（プラグイン）をインストール
./	nexacro14_client_license.xml	製品ライセンスファイル
./Test/	SampleADL.xadl.js	ADL（JavaScript）
./Test/	start.json	Bootstrap file
./Test/	[nexacro14lib]	コンポーネントライブラリフォルダ（JavaScript）
./Test/	[_theme_]	テーマフォルダ（JavaScript）
./Test/	[Base]	アプリケーションフォルダ（JavaScript）

> 配布に必要なファイルは、「Generate」ではなく「Deploy」機能として作成する必要があります。nexacro studioの[Build>Deploy]メニューから必要な項目を選択して、配布ファイルを生成します。
> 「Deploy」オプションの中で「Merge JSON file」の項目は、必須選択項目です。

nexacro platformランタイムエンジンのインストール時に、ブラウザごとに配布方法が異なるため、以下のように使用するブラウザを確認し、適切なページに移動します。

```
<html>
<head>
<script language="javascript">
function fn_onload()
{
    if ((navigator.userAgent.indexOf("MSIE") > -1) ||
        ( navigator.userAgent.indexOf("Trident") != -1)) {
        window.location.href = "./nexacroAX.html";
    } else {
        window.location.href = "./nexacroPlugin.html";
    }
}
</script>
</head>
<body onload="fn_onload()">
</body>
</html>
```

> インストールと実行ページの場所とページ移動方式は、OS環境によって異なる場合があります。

2.1.1 配布ページ

配布ページはnexacro platformエンジンを実行するために、ウェブブラウザが最初に実行するHTMLファイルです。ファイル名は、開発時に自由に指定することができます。

nexacroAX.html

```
<HTML>
<HEAD>
<meta http-equiv="content-type" content="text/html;charset=UTF-8"/>
<TITLE> Web Browser Embed nexacro platform Sample Page </TITLE>
<SCRIPT LANGUAGE="javascript">
function fn_load()
{
    // Bootstrapパスを指定します。
    nexacroAXCtrl.bjson = "./Test/start.json";
    // サービスKey値を設定します。
    // url上のサービスの唯一の値である必要があります。
    nexacroAXCtrl.key = "nexacro_Sample";
    // nexacro platformエンジンを実行します。
    nexacroAXCtrl.run();
}
</SCRIPT>
</HEAD>
<BODY onload="fn_load()">
<SCRIPT LANGUAGE="javascript">
document.write('<OBJECT ID="nexacroAXCtrl" width="50%" height="50%" CLASSID="
CLSID:3CF3951B-34BD-4E22-933B-AE2BE0FA8941" CodeBase="./nexacro14_SetupEngine_N
.cab#VERSION=2014,7,11,2"/>');
</SCRIPT>
</BODY>
</HTML>
```

エンジンが最初にインストールされると、ウェブブラウザにインストールメッセージが表示され、ユーザーがインストールボタンをクリックすると、インストールが実行されます。

nexacroPlugin.html

インターネットエクスプローラではない場合には、プラグインをインストールするためのスクリプトを記述する必要があります。次のコードのいずれExistPlugin関数は、プラグインがインストールされたのかを確認し、インストールされていなければ、インストールファイルをダウンロードしてインストールするように説

明します。

```
<HTML>
<HEAD>
<meta http-equiv="content-type" content="text/html;charset=UTF-8"/>
<TITLE> Web Browser Embed nexacro platform Sample Page </TITLE>
<SCRIPT LANGUAGE="javascript">
function fn_load()
{
    if (ExistPlugin() == false)
    {
        shtml = '<p><ul>';
        shtml += '<li>がコンテンツを表示するために使用するプラグインがありません。</li>';
        shtml += '<li>プラグインをインストールするためには<a href="./nexacro14_
SetupEngine_N.exe">[nexacro14_SetupEngine_N.exe]</a>
をダウンロードした実行してください。</li>';
        shtml += '</ul></p>';
        document.writeln(shtml);
    }
    else
    {
        nexacroplugin.bjson = "./Test/start.json";
        nexacroplugin.key = "nexacro_Sample";
        nexacroplugin.launch();
    }
}
function ExistPlugin()
{
    var mimetype = navigator.mimeTypes["application/nexacro14-plugin"];
    if (mimetype)
    {
        var enablePlugin = mimetype.enabledPlugin;
        if (enablePlugin)
            return true;
        else
            return false;
    }
    else
    {
        return false;
    }
```

```
}
</SCRIPT>
</HEAD>
<BODY onload="fn_load()">
<SCRIPT LANGUAGE="javascript">
document.write('<EMBED ID="nexacroAXCtrl" type="application/nexacro14-plugin"
width="50%" height="50%"/>');
</SCRIPT>
</BODY>
</HTML>
```

2.1.2 SampleADL.xadlファイル

SampleADL.xadlは、サンプル応用プログラムを実行するための応用プログラムの配布及び実行環境の設定情報を格納しています。このファイルはnexacro studioにより管理されます。

以下はSampleADL.xadlの例題です。

```
<?xml version="1.0" encoding="utf-8"?>
<ADL version="1.2">
<TypeDefinition url="default_typedef.xml"/> ❶
<GlobalVariables url="globalvars.xml"/> ❷
<Application id="nexacro_Sample" codepage="utf-8"
     licenseurl="./nexacro14_client_license.xml" …> ❸
     <Layout> ❹
          <MainFrame id="mainframe" …>
               <ChildFrame formurl="Base::SampleForm.xfdl"…/> ❺
          </MainFrame>
     </Layout>
    <ScreenInfo/>
</Application>
<Script><![CDATA[ ]]></Script>
</ADL>
```

上記のファイルにより次の内容がわかります。

1. コンポーネントの配布情報がdefault_typedef.xmlに定義されていることを示します。

2. 応用プログラムで使用するグローバル変数がglobalvars.xmlに保存されていることを示します。
3. 応用プログラムのIDがnexacro_Sampleであることが確認できます。
4. アプリケーションのモデルは、MainFrameの下にChildFrameがある単一のフレーム形状です。
5. 最初の画面に表示されるフォーム画面が"SampleForm.xfdl"であることが確認できます。

> この他にも、様々な応用プログラムの環境についての情報があります。このファイルはnexacro studioが管理するファイルであるため、直接に編集しないことをお勧めします。

> ライセンスファイルは、製品を購入する時に提供され、任意に変更すると、アプリケーションが正常に動作しないことがあります。

2.1.3 default_typedef.xmlファイル

default_typedef.xmlは、サンプル応用プログラムの実行のためのコンポーネントの配布情報を格納しています。また、開発者が開発したコンポーネントを配布する際にも使用されます。

以下はdefault_typedef.xmlの例題です。

```
<?xml version="1.0" encoding="utf-8"?>
<TypeDefinition version="1.1">
  <Modules defaultpath="">
    <Module url="CompBase.json"/>
    <Module url="ComComp.json"/>
    <Module url="Grid.json"/>
    <Module url="DeviceAPI.json"/>
  </Modules>
  <Components>
    <Component type="JavaScript" id="Div" classname="nexacro.Div"/>
    <Component type="JavaScript" id="Button" classname="nexacro.Button"/> ❶
    <Component type="JavaScript" id="PopupDiv" classname="nexacro.PopupDiv"/>
    <Component type="JavaScript" id="Combo" classname="nexacro.Combo"/>
    <Component type="JavaScript" id="CheckBox" classname="nexacro.CheckBox"/>
...
</Components>
  <Services>
```

```
    <Service prefixid="Base" type="form" url="./Base/"
       version="0" communicationversion="0"/> ❷
  </Services>
</TypeDefinition>
```

上記のファイルを通じて、次の内容を分かられます。

1. Buttonコンポーネントの配布情報が含まれています。

   詳細は、次のようです。

   配布されるコンポーネントは、Buttonコンポーネントです。

   Buttonコンポーネントを使用するためにはnexacro.Buttonとして定義されたJavaScriptモジュール(Button.js)が必要です。

2. Form urlのためのサービスグループを定義しました。

   以来、Form名を指定するときに、「Base/Form名」を指定すると、urlが「http://127.0.0.1/Test/Base/Form名」に置き換えられます。

> このファイルはnexacro studioに自動生成されるファイルであるため、直接に編集しないことをお勧めします。

2.2 Launcher配布

Launcher配布は、Webブラウザを使用せずに、nexacro platformランタイムでのみアプリケーションを実行するときに使用します。exe配布と異なる点は、nexacro platformランタイムを直接実行せずに、Launcherを利用して実行します。

サービス管理者は、配布時、下記のようなディレクトリ構造で管理することができます。

Path	File / Folder	説明
./	SampleStart.html	インターネットエクスプローラの使用時のnexacro platformエンジンのインストールページ
./	SampleLaunch.html	インターネットエクスプローラの使用時のnexacro platformアプリケーションの実行ページ
./	nexacro14_Laun	インターネットエクスプローラの使用時のnexacro platformランチャー（Act

Path	File / Folder	説明
	cher_N.cab	ive X）インストール
./	nexacro14_SetupEngine_N.cab	インターネットエクスプローラの使用時のnexacro platformランタイムエンジン（Active X）をインストール
./	nexacro14_client_license.xml	製品ライセンスファイル
./Test/	SampleADL.xadl.js	ADL（JavaScript）
./Test/	start.json	Bootstrap file
./Test/	[nexacro14lib]	コンポーネントライブラリフォルダ（JavaScript）
./Test/	[_theme_]	テーマフォルダ（JavaScript）
./Test/	[Base]	アプリケーションフォルダ（JavaScript）

2.2.1 SampleStart.htmlファイル

Plugin配布と異なり、このファイルはnexacro platformの設置のみのために使用される HTMLファイルです。このファイルのファイル名は、開発者が自由に指定することができます。

以下の内容はこのファイルの例題です。

```
<HTML>
<HEAD>
<TITLE> nexacro platform Install Sample Page </TITLE>
</HEAD>
<SCRIPT LANGUAGE=javascript FOR=nexacroAXCtrl EVENT="error(errcode,errmsg)">
        alert("errcode: " + errcode + "\nerrmsg: " + errmsg);
</SCRIPT>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
document.write('<OBJECT ID="nexacroAXCtrl" '
+ 'CLASSID="CLSID:3CF3951B-34BD-4E22-933B-AE2BE0FA8941" width="0" height="0" '
+ '"CODEBASE="./nexacro14_SetupEngine_N.cab#VERSION=2014,7,11,2"  '
+ 'onError="fn_onError()"> '
+ '</OBJECT>');
</SCRIPT>
</BODY>
```

```
</HTML>
```

特別な実行命令はなく、<SCRIPT LANGUAGE="JavaScript"> ~ </SCRIPT>までのみ登録すればnexacro platformエンジンが設置されます。

2.2.2 SampleLauch.htmlファイル

SampleLaunch.htmlは、Launcherを駆動するために実行するHTMLファイルです。このファイルのファイル名は、開発者が自由に指定することができます。

以下の内容はこのファイルの例題です。

```
<HTML>
<HEAD>
<TITLE> nexacroLauncher Sample page </TITLE>
<SCRIPT LANGUAGE="javascript">
function fn_load ()
{
    nexacroLauncher.bjson = "./Test/start.json"; ❷
    nexacroLauncher.key = "nexacro_Sample"; ❸
    nexacroLauncher.launch(); ❹
}
</SCRIPT>
</HEAD>
<SCRIPT LANGUAGE=javascript FOR=nexacroLauncher EVENT="error(errcode,errmsg)">
        alert("errcode: " + errcode + "\nerrmsg: " + errmsg);
</SCRIPT>
<BODY onload="fn_load()">
<SCRIPT LANGUAGE="javascript">
document.write('<OBJECT ID="nexacroLauncher" '
+ 'CLASSID="CLSID:812CB626-69B7-4FEB-B42F-E3BBFF53E23B" width="90%" height=
"90%" '
+ 'CODEBASE="./nexacro14_Launcher_N.cab#VERSION=2014,7,11,2"  '
+ 'onError="fn_onError()">' ❶
+ '</OBJECT>');
</SCRIPT>
</BODY>
</HTML>
```

上記のファイルにより次の内容が確認できます。

1. Launcherを設置します。
2. サービスのキー値を設定します。

   (Plugin配布時、nexacroAX.htmlファイルに指定されたkey設定値と同じ役割をします。)
3. Bootstrapパスを指定します。
4. Launcherによりnexacro platformエンジンを実行します。

2.2.3 SampleLaunch.htmlとSampleStart.htmlの統合

前で説明したSampleLaunch.htmlとSampleStart.htmlは、1つのファイルとして統合することができます。

以下は統合したファイルの例題です。

```
<HTML>
<HEAD>
<TITLE> nexacro platform Launcher execution </TITLE>
<SCRIPT LANGUAGE="javascript">
function fn_load ()
{
    nexacroLauncher.bjson = "./Test/start.json";
    nexacroLauncher.key = "nexacro_Sample";
    nexacroLauncher.launch(); ❸
}
</SCRIPT>
</HEAD>
<SCRIPT LANGUAGE=javascript FOR=nexacroAXCtrl EVENT="error(errcode,errmsg)">
        alert("errcode: " + errcode + "\nerrmsg: " + errmsg);
</SCRIPT>
<SCRIPT LANGUAGE=javascript FOR=nexacroLauncher EVENT="error(errcode,errmsg)">
        alert("errcode: " + errcode + "\nerrmsg: " + errmsg);
</SCRIPT>
<BODY onload=" fn_load()">
<SCRIPT LANGUAGE="javascript">
document.write('<OBJECT ID="nexacroAXCtrl" '
+ 'CLASSID="CLSID:3CF3951B-34BD-4E22-933B-AE2BE0FA8941" width="0" height="0" '
+ '"CODEBASE="./nexacro14_SetupEngine_N.cab#VERSION=2014,7,11,2" >'
```

```
+ '</OBJECT>'); ❶
document.write('<OBJECT ID="nexacroLauncher" '
+ 'CLASSID="CLSID:812CB626-69B7-4FEB-B42F-E3BBFF53E23B" width="0" height="0" '
+ '"CODEBASE="./nexacro14_Launcher_N.cab#VERSION=2014,7,11,2" >'
+ '</OBJECT>'); ❷
</SCRIPT>
</BODY>
</HTML>
```

1. nexacro platformエンジンを設置します。
2. Launcherを設置します。
3. Launcherを実行します。

2.3 exe配布

exe配布はcabファイルを使用せずにnexacro platformエンジンを手動でインストールした後、アプリケーションを実行する場合に使用します。

通常のCDでアプリケーションを配布したり、インストールファイルをダウンロードする場合によく使われる方法です。但し、画面出力のためのnexacro platformコンポーネントは、HTTPサーバーを使用して配布されるため、別のHTTPサーバーを運用する必要があります。

Context root		
	SampleADL.xadl.js	ADL (JavaScript)
	start.json	Bootstrap file
	[コンポーネント]	コンポーネントライブラリフォルダ (JavaScript)
	[_theme_]	テーマフォルダ(JavaScript)
	[サービスグループ]	アプリケーションフォルダ(JavaScript)

2.3.1 コマンドプロンプトでnexacro platformを実行

MS Windowsのコマンドプロンプトでnexacro platformエンジンを直接に実行します。

コマンドプロンプトで以下の通り入力すると、Launcherによりnexacro platformエンジンを起動させるのと同じです。

```
"C:\Program Files\nexacro\14\nexacro.exe" -K "nexacro_Sample" -S "http
://127.0.0.1/Test/start.json"
```

-K

Key値を指定します。Plugin配布時、SampleStart.htmlファイルに指定されたkey設定値と同じ役割をします。

-S

Bootstarpパスを指定します。

2.4 ショートカットによる配布

この方法は、初めてウェブサーバーにより接近した使用者に“短縮アイコン”を生成し、その後、応用プログラムを実行する際に“短縮アイコン”を使用する方法で、“Launcher配布”と“exe配布”をミックスした形態です。

Context root		
	MakeShortcut.html	MakeShortcut.htmlファイル 参照
	SampleIconImage.ico	
	nexacro14_SetupEngine_N.cab	
	SampleADL.xadl.js	ADL (JavaScript)
	start.json	Bootstrap file
	[コンポーネント]	コンポーネントライブラリフォルダ(JavaScript)
	[_theme_]	テーマフォルダJavaScript)
	[サービスグループ]	アプリケーションフォルダJavaScript)

2.4.1 MakeShortcut.htmlファイル

MakeShortcut.htmlは、nexacro platformエンジンを設置し応用プログラムを実行するためのショートカットを生成する内容を含んでいます。このファイルのファイル名は、開発者が自由に指定することができます。

以下の内容はこのファイルの例題です。

```
<HTML>
<HEAD>
<TITLE>Make shortcut Sample Page </TITLE>
<SCRIPT LANGUAGE="javascript">
function fn_make_shortcut()
{
     nexacroLauncher.key = "nexacro_Sample";
     nexacroLauncher.bjson = " ./Test/start.json";
     nexacroLauncher.makeshortcut("shortcut",
          "./SampleIconImage.ico",
          "desktop", false);
}
</SCRIPT>
</HEAD>
<BODY onload="fn_make_shortcut()">
<SCRIPT LANGUAGE="javascript">
document.write('<OBJECT ID="nexacroAXCtrl" '
+ 'CLASSID="CLSID:3CF3951B-34BD-4E22-933B-AE2BE0FA8941" width="90%" height
="90%" '
+ '"CODEBASE="./nexacro14_SetupEngine_N.cab#VERSION=2014,7,11,2" >'
+ '</OBJECT>');
document.write('<OBJECT ID="nexacroLauncher" '
+ 'CLASSID="CLSID:812CB626-69B7-4FEB-B42F-E3BBFF53E23B" '
+ '"CODEBASE="./nexacro14_SetupEngine_N.cab#VERSION=2014,7,11,2" >'
+ '</OBJECT>');
</SCRIPT>
</BODY>
</HTML>
```

このページにアクセスしたウェブブラウザのユーザの端末機には、"shortcut"というショートカットが生成され、そのショートカットを実行するとnexacro platformエンジンが実行されます。

2.5 サードパーティモジュールの配布

nexacro platformランタイムでサードパーティの製品を使用する場合には、該当の製品をCabファイルで製作しなければなりません。

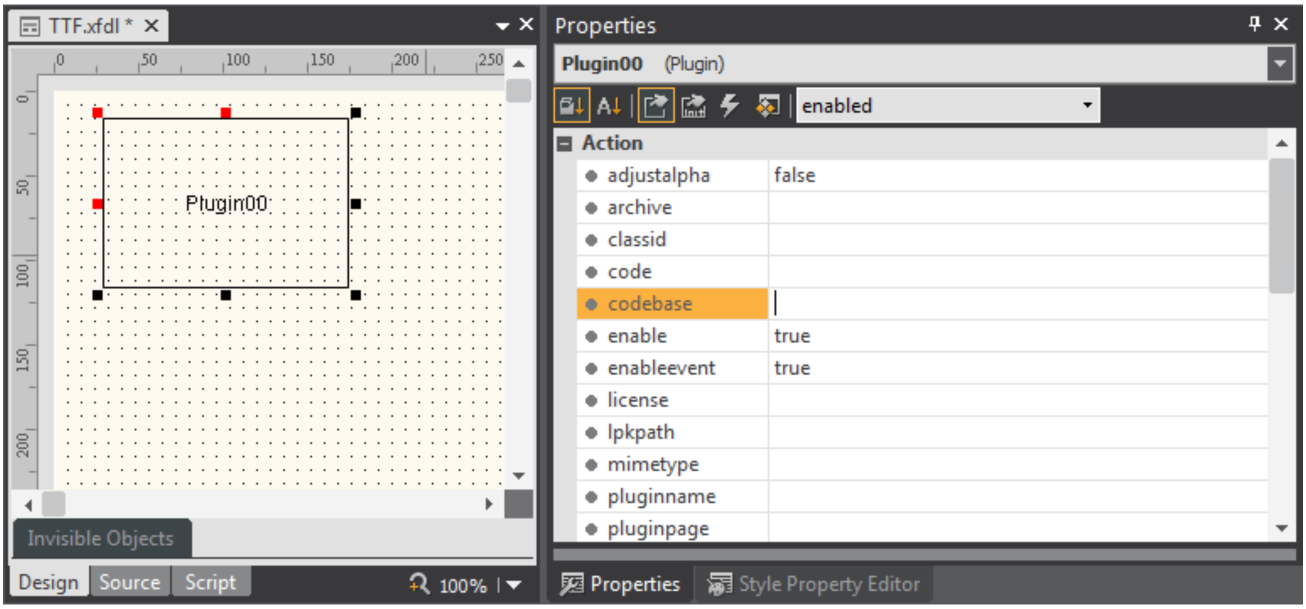
2.5.1 サードパーティモジュールの製作

nexacro platformはウェブブラウザが採用するCab設置方法を採用するため、ウェブブラウザで設置及び実行されるよう製作されたCabファイルはnexacro platformでも設置及び実行ができます。

> しかし、必公認された機関の認証を受けたCabファイルの場合にのみ使用できますので、ご留意ください。

2.5.2 サードパーティモジュールの使用

Cabファイルで製作されたサードパーティモジュールは使用が簡単です。nexacro studioでフォームの開発時にOCXコンポーネントを使用し、該当コンポーネントのcodebaseプロパティにCabファイルのURLのみを登録します。



第3章

主要モジュールの実行方法

nexacro platformが提供する3つのモジュールは、使用者がnexacro platform以外の他のプログラムで実行します。この中の2つ(NexacroAX, NexacroLauncherAX)はウェブブラウザにより実行され、残りの1つ(nexacro.exe)はOSのコマンドプロンプトで実行されます。

3.1 NexacroAX

NexacroAXは、nexacro platformエンジンをウェブブラウザ上で実行させるためのActiveXモジュールです。

このモジュールは、以下のセットアップファイルに含まれています。

- nexacro14_SetupDeveloper_N.exe
- nexacro14_SetupEngine_N.exe
- nexacro14_SetupEngine_N.cab

NexacroAXは、ウェブブラウザ上のオブジェクトであるため element(プロパティ、メソッド、イベント)を持ちます。ここではelementの概略的な説明のみをします。

3.1.1 クラスID

AcitveXのためのクラスID“812CB626-69B7-4FEB-B42F-E3BBFF53E23B”です。

以下は、このモジュールを活性化するためのHTMLコードです。

```
<SCRIPT LANGUAGE="javascript">
document.write('<OBJECT ID="nexacroLauncher" '
+ 'CLASSID="CLSID:812CB626-69B7-4FEB-B42F-E3BBFF53E23B"'
+ '"CODEBASE="./nexacro14_Launcher_N.cab#VERSION=2014,5,27,1" '
+ 'onError="fn_onError()" >'
+ '</OBJECT>');
```

3.1.2 プロパティ

NexacroAXのプロパティを整理すると、以下の通りです。必須項目された値を指定すればnexacro platformエンジンを起動することができます。

プロパティ名	必須	説明 / 基本値
key	O	サービスのキー値を設定します。 同一ADL上で唯一な値でなければなりません。
		基本値なし
xadl	O	変換されたサービスADLJavaScriptファイルのパスを指定します。
		基本値なし
componentpath	X	使用者のPC上にコンポーネントが保存されるパスを指定します。一般的には、この値は指定しません。
		基本値 ="%USERAPP%\nexacro\14\Component\" %USERAPP%の値は“付録A. エイリアスパス”を参照してください。
autosize	X	nexacro platformのサイズがウェブブラウザのサイズの影響を受けて動的に変わるのか否かを指定します。 falseの場合には、ウェブブラウザのPlugin WidthとHeight値が適用されます。
		基本値=true
commthreadcount	X	nexacro platformエンジンが使用する通信スレッドの個数を指定します。 使用者の端末機にCPUコアが複数ある時には、値を大きく指定してください。 この機能は通信処理にのみ使用するため、いくつかの通信を同時に使用したい場合には、値を大きくすればスピーディーな応答結果が得られます。 しかし、ネットワーク自体が遅い場合には、値を大きくしてもあまり効果がありません。
		基本値=5
errorfile	X	nexacro platformエンジンが表示するエラー情報が定義されたファイルパスです。

プロパティ名	必須	説明
		基本値
		指定しない場合には、nexacro platformエンジンの基本エラー情報が表示されます。一般的には指定しません。
		基本値なし
handleflag	X	nexacro platform応用プログラムが終了してから、exit()イベントを発生させるか否かを指定します。 指定しない場合にはtrueとなりexit()が発生します。
		基本値=true
Commthreadwaittime	X	nexacro platform応用プログラムが使用するスレッドが何もしない場合には、該当のスレッドが終了します。この時の終了前の待ち時間(秒)を設定するプロパティです。
usewininet	X	サーバーとの通信方法を指定するプロパティとして、WININET通信ライブラリを使用して通信をするかを設定するプロパティです。
		基本値=false
autostart	X	NexacroAX コントロールが作成される時に自動的にrun()を 実行するかどうかを決定するプロパティです。
		基本値=false

3.1.3 メソッド

NexacroAXを実行することができるメソッドを概略的に整理すると、以下の通りです。

イベント名	説明
callscript	NexacroAXで起動中のnexacro platformエンジンのスクリプトメソッドを実行するためのメソッドです。
getvariablevalue	NexacroAXでnexacro platformエンジンのグローバル変数の変数値を得るメソッドです。
run	NexacroAXがnexacro platformエンジンを実行するメソッドです。
setvariablevalue	NexacroAXでnexacro platformエンジンのグローバル変数の変数値を設定するメソッドです。
stop	PlugInがnexacro platformの全てのマネージャを中止させるメソッドです。
quit	PlugInがを終了するメソッドです。
addWebInfo	現在のブラウザのクッキーの情報を自動的にglobalvariableのcookie変数に追加するメソッドです。

3.1.4 イベント

NexacroAXで発生するイベントを整理すると、以下の通りです。

イベント名	説明
addlog	• nexacro platformスクリプトでtrace()関数を実行した時に発生するイベントです。 • nexacro platformスクリプトでtrace()関数引数値がそのままイベントの引数値として適用されます。
beforeexit	• nexacro platformエンジンの終了直前に発生するイベントです。 • Beforeexitイベント後にexitイベントが発生します。 BeforeexitイベントでhandledflagИ値をfalseで設定すると、exitイベントが発生しません。即ち、終了しません。 • nexacro platformスクリプトでexit関数を呼び出した時に発生し、アプリケーションのonbeforeexitイベントの発生後にNexacroAXでbeforeexitが発生します。
communication	nexacro platformの通信を開始する時と終了する時に発生するイベントです。
error	NexacroAXのロード中またはnexacro platformエンジンの実行中にエラーが生じると発生するイベントです。
exit	• nexacro platformエンジンが終了する時に発生するイベントです。 • beforeexit()イベントの影響を受けます。
load	• nexacro platformエンジンの初期のロードが完了してから発生するイベントです。 • nexacro platformスクリプトではonloadイベントが発生します。 • loadtypedefinition loadingglobalvariables ロードの順にイベントが発生します。
loadingglobalvariables	• グローバル変数のオブジェクトが生成されてから発生するイベントです。 データセットとフォームのURLはロード前であるため、イベント内でロードに必要な情報を追加・修正すれば変更された値でロードされます。 • nexacro platformスクリプトでもアプリケーションのonloadingglobalvariablesイベントが発生します。
loadtypedefinition	• TypeDefinitionに登録されたコンポーネントのロードを完了してから発生するイベントです。 コンポーネントの配布も完了したことを意味します。 • nexacro platformスクリプトでもアプリケーションのonloadtypedefinitionイベントが発生します。
usernotify	• nexacro platformスクリプトでuserNotify()関数を呼び出す場合にPlugInで発生するイベントです。 • nexacro platformスクリプトでのuserNotify()関数の引数値がそのままイベントの引数値として適用されます。

3.2 Launcher

LauncherはInexacro platform Engineをnexacro platformランナー(runner)で実行するためのActiveXモジュールです。

このモジュールは、以下のセットアップファイルに含まれています。

- nexacro14_SetupDeveloper_N.exe
- nexacro14_SetupEngine_N.exe
- nexacro14_SetupEngine_N.cab
- nexacro14_Launcher_N.cab

Launcherは、ウェブブラウザ上のオブジェクトであるためelement(プロパティ、メソッド、イベント)を持ちます。

ここではelementの概略的な説明のみをします。

3.2.1 クラスID

AcitveXのためのクラスIDは“812CB626-69B7-4FEB-B42F-E3BBFF53E23B”です。

以下は、このモジュールを活性化するためのHTMLコードです。

```
<SCRIPT LANGUAGE="javascript">
document.write('<OBJECT ID="nexacroLauncher" '
+ 'CLASSID="CLSID:812CB626-69B7-4FEB-B42F-E3BBFF53E23B" '
+ '"CODEBASE="./nexacro14_Launcher_N.cab#VERSION=2014,7,11,2" '
+ 'onError="fn_onError()">'
+ '</OBJECT>');
```

3.2.2 プロパティ

Launcherのプロパティを整理すると、以下の通りです。必須項目を指定すればnexacro platformエンジン

を起動させることができます。

プロパティ名	必須	説明 基本値
Key	O	サービスのキー値を設定します。 同一ADL上で唯一な値でなければなりません。 基本値なし
Xadl	O	変換されたサービスADLJavaScriptファイルのパスを指定します。 基本値なし
commthreadcount	X	nexacro platformエンジンが使用する通信スレッドの個数を指定します。 使用者の端末機にCPUコアがいくつかある時には、値を大きく指定してください。 この機能は通信処理にのみ使用するため、いくつかの通信を同時に使用したい場合には、値を大きくすればスピーディーな応答結果が得られます。しかし、ネットワーク自体が遅い場合には、値を大きくしてもあまり効果がありません。 基本値=5
componentpath	X	使用者のPC上にコンポーネントが保存されるパスを指定します。 一般的には、この値は指定しません。 基本値 ="%USERAPP%\nexacro\14\Component\" %USERAPP%の値は"付録A. エイリアスパス"を参照してください。
errorfile	X	nexacro platformエンジンが表示するエラー情報が定義されたファイルパスです。指定しない場合には、nexacro platformエンジンの基本エラー情報が表示されます。 一般的には指定しません。 基本値なし
globalvalue	X	nexacro platformエンジンが駆動する時にグローバル変数に追加される変数を設定するプロパティです。 基本値なし
loadingimage	X	nexacro platformエンジンが画面やデータのロード時に表示するイメージを設定するプロパティです。 サーバーパスが入力される場合には、直接にダウンロードしてから、設定されたコンポーネントパスの下に保存します。 基本値なし
splashimage	X	nexacro platformエンジンがロードされるまで表示するスプラッシュイメージのパスを指定します。 指定しない場合には、nexacro platformの基本イメージが表示されます。 このイメージは画面の中央に表示されます。

プロパティ名	必須	説明 基本値
		このイメージは、nexacro platformエンジンがロードされる間のみ表示されます。 基本値なし
onlyone		キーとXADLの値を、同一インスタンス1つだけ使用するかどうかを設定するプロパティです。 基本値=false
commthreadwaittime		nexacro platform応用プログラムが使用するスレッドが何もしない場合には、該当のスレッドが終了します。この時の終了前の待ち時間(秒)を設定するプロパティです。
enginesetupkey		実行パスとして使用する特定バージョンの ProductKeyの値を指定するプロパティです。 サイトで別途作成したSetup Moduleを使用する時、このプロパティに Setupを生成する時に与えられる Product Keyを設定すれば固定されたパスにより、LauncherでRuntimeの実行が不可だった問題を解決することができます。

3.2.3 メソッド

Launcherを実行することができるメソッドを概略的に整理すると、以下の通りです。

イベント名	説明
launch	nexacro platformエンジンを専用のブラウザで実行させます。
makeshortcut	ショートカットを生成します。
isexistshortcut	入力された引数に対応するショートカットアイコンがあるかどうかを確認するメソッドです。
getEngineVersion	システムにインストールされたエンジンキーに対応するエンジンインストールバージョンをインポートするメソッドです。
download	アプリケーション実行時に使用されるリソースをダウンロードするメソッドです。
deleteshortcut	ショートカットアイコンを削除するメソッドです。
addWebInfo	現在のブラウザのクッキーの情報を自動的にglobalvariableのcookie変数に追加するメソッドです。

3.2.4 イベント

Launcherで発生するイベントを整理すると、以下の通りです。

イベント名	説明
error	Launcherの実行中にエラーが発生すると、実行されるイベントです。

3.3 nexacro.exe

nexacro platformのエンジンは、MS Windows上のコマンドプロンプトで直接に実行することができます。主にショートカットを使用しますが、実行ファイルのnexacro.exeを様々なオプションにより実行することができます。

```
nexacro.exe –option1 option1_value –option2 option2_value … -optionN –optionN_
value
```

3.3.1 実行オプション

以下はオプションの説明です。

実行 オプション	値	説明
X (必須)	パス名(string)	変換されたサービスADL JavaScriptファイルのパスを指定します。
K (必須)	キー名(string)	サービスのキーを設定します。 URL上のサービスの中で唯一の値でなければなりません。
C	パス名(string)	使用者の端末機にコンポーネントが設置されるパスを指定します。 指定しなければ、 "%USERAPP%\nexacro\14\Component\"が指定されます。 %USERAPP%の値は“付録. エイリアスパス”を参照してください。
SP	パス名(string)	スプラッシュイメージのパスを指定します。

実行 オプション	値	説明
		指定しない場合には、nexacro platformの基本イメージが表示されます。 このイメージは画面の中央に表示されます。 このイメージは、nexacro platformエンジンがロードされる間のみ表示されます。
LF	位置値(int)	アプリケーションの初期実行の左位置を指定します。 最初の画面のメインフレームフレームに適用されます。
TP	位置値(int)	アプリケーションの初期実行の上部位置を指定します。 最初の画面のメインフレームフレームに適用されます。
WD	サイズ(int)	アプリケーションの初期実行の幅を指定します。 最初の画面のメインフレームフレームに適用されます。
HT	サイズ(int)	アプリケーションの初期実行の高さを指定します。 最初の画面のメインフレームフレームに適用されます。
ERR	パス名 (string)	エラーコード値を定義した“errordefine.xml”ファイルのパスを指定します。 nexacro platformエンジンが表示するエラー情報を変更する時に使用します。 指定しない場合には、nexacro platformエンジンの基本エラー情報が表示されます。
THR	個数(int)	エンジン内部の通信及びその他のマネージャのスレッド数を指定します。 指定しない場合には5となります。 使用者の端末機が旧仕様の場合には、値を大きく指定してください。
TW	時間(int)	nexacro platformランタイムの通信スレッドの待ち時間を指定します。
LG	パス名 (string)	LoadingImageを設定することができるパスを指定します。
OO	多重起動 (bool)	同一なキー、XADLとしてXREを起動させる時、XREを多重起動させるかどうかを指定します。
GF	ファイル名 (string)	Launcherからグローバル変数を設定した場合、nexacro platformエンジンに値を渡すために使用するTempファイル名です。 nexacro platformが立ち上がる時、読まれて削除します。 Launcherを使用せず直接入力する場合には正常に動作しない場合があります。
V	Variable 文字列 (string)	XREを開く時グローバル変数に追加する変数リストを設定します。 変数名：タイプ＝値　 変数名：タイプ＝値　の形式に入力します。 区切りはスペースです。 つまり、"A : s=b　C : g=d　E=f"の形式になります。

実行 オプション	値	説明
		通信時、クッキーとして使用する変数はタイプに s、通常の変数は型を入力しないか、または gで表記します。
SIE	可否 (bool)	Adlのロード時にエラーが発生する場合、エラーメッセージボックスを表示させて終了するかどうかを指定します。
CT	時間秒(int)	最初のネットワークが接続されているかチェックするTimeoutを設定します。ネットワークが接続されているかどうかをApplicationが起動される前にチェックします。Timeout時間の間にネットワークが接続されていない場合、自動的に終了します。
SCR	スクリーン ID 文字列(string)	LayoutManagerで使用するための ScreenIDを設定します。

3.3.2 nexacro.exeファイルのパス

nexacro.exeファイルは、“nexacro14_SetupEngine_N.cab”または“nexacro14_SetupEngine_N.exe”で設置されます。

設置パスは、MS Windowsの場合には以下の通りです。

```
%PROGRAMFILES%\nexacro\14\nexacro.exe
```

第III部

Runtime for iOS/Android

第1章

アプリ開発の概要

nexacro platformは、同じ開発環境で、1つのソースコードで、様々なオペレーティング·システムと使用環境をサポートするアプリケーションを開発することができます。ただし、実際のユーザーに配信されるアプリケーションの形で配布される過程で、使用環境によって、一部機能と配布環境が異なることになります。

この章では、スマートフォンやタブレット機器のいずれiOSとAndroid OSを対象に、開発者またはシステム管理者が必要な内容を説明しています。

1.1　技術構成要素

iOSとAndroid OSに配布されるアプリケーションは、移動中に使用し、さまざまな内蔵センサー機能を活用することができ、スマートフォンの場合には、電話や文字メッセージ機能を連動することができます。これにより、Device API、開発、実行、配布についての理解が必要になります。

1.1.1　Device API

次は、nexacro platformアプリで提供される Device APIです。

Device API	機能内容
System	端末の基本情報提供 ダイヤル機能提供 簡単なサウンドファイル再生機能提供 • Objects > Misc. Objects > System
Acceleration	加速度計センサーを使って端末の加速度情報を提供
LiteDB	ローカル DBを使用するための機能を提供

Device API	機能内容
VirtualFile	ファイル入出力のための機能を提供
AudioPlayer	サウンドファイルの再生コントロールのための機能を提供
AudioRecorder	オーディオレコーディングのための機能を提供
Camera	端末の内蔵カメラを利用できる機能を提供
ContactSet	アドレス帳の連絡先の検索/追加/削除/修正機能を提供
ExternalAPI	外部機器へアクセスまたは連動するための機能を提供
GeoLocation	GPS. WPSを使って端末の位置情報を提供 GPS - Global Positioning System WPS - WiFi Positioning System
ImagePicker	端末の写真アルバムにアクセスできる機能を提供
Map	マップ表示のためのMapコンポーネントを提供
Network	端末のネットワークステータス (No Network、3G、Wifi)を確認する機能を提供 特定のサイトへの通信が可能かどうかを確認する機能を提供
Sms	SMS転送機能提供 SMS受信検知機能提供 受信された SMSライブラリの検索/削除機能を提供
Vibrator	振動を発生させられる機能を提供

提供される機能は、製品のバージョンによって、追加されたり変更されることがあり、端末のOSの特性上、サポートされる機能の部分的な制限がある場合があります。詳しくは、nexacro studioに含まれているリファレンスガイドを参照してください。

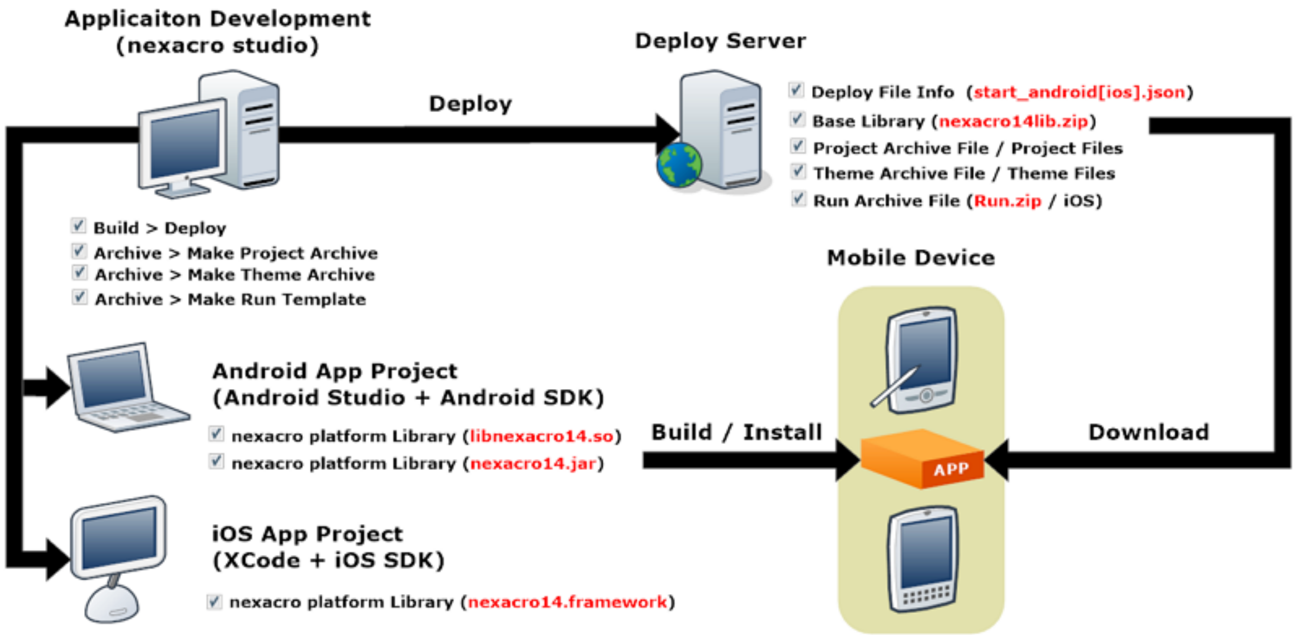
1.1.2 開発環境

画面とサービスを開発する方式は、ランタイムやHTML5が共に同じですが、iOSとAndroid OSで使用できるDevice APIを追加コンポーネントとして活用できるということが違います。生成されたアプリケーションの基本的なテストは、nexacro studio内で実行してみることができますが、Device APIの実際の動作は、機器にアプリを配布して直接テストしてみる必要があります。

Android、iOSでは、任意のプログラムを直接実行することができず、アプリ形で作られたアプリケーションのみを実行することができます。したがって、nexacro platformをベースに開発されたアプリケーションを、ウェブブラウザではなく、端末から直接実行するには、アプリ形で開発、配布する必要があります。

nexacro platformは、Android、iOS開発環境に必要なライブラリを提供して簡単にアプリを作成できるようにサポートしており、パッケージ（アーカイブファイルの作成）をベースに作られたアプリケーションを配布することができます。

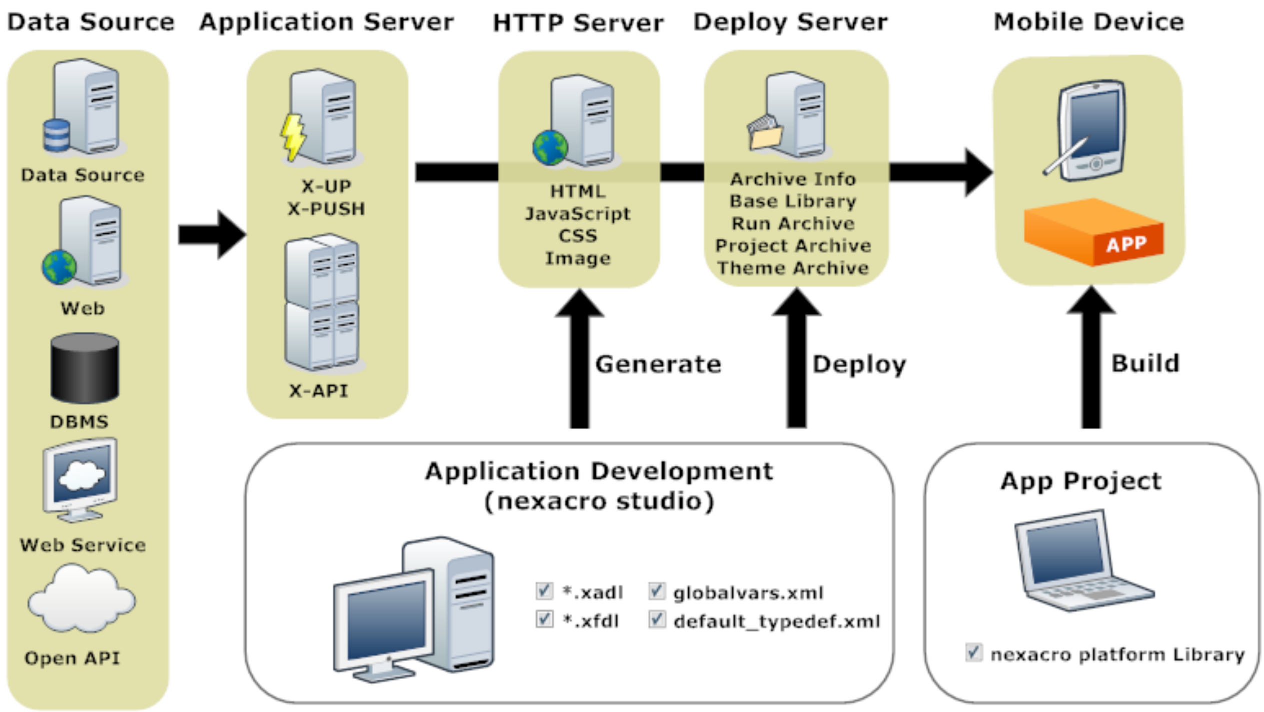
iOS、Androidで動作するnexacro platformベースのアプリを開発するプロセスは以下のとおりです。



> nexacro studioでマーケットに配布することができるアプリを作成したり、シミュレータでテストする直接的な方法は、別途提供されません。

1.1.3 アプリケーションの実行環境

nexacro platformで開発されたアプリは、指定された配備サーバーのstart_android[ios].jsonファイルに設定された情報によって必要なファイルを端末にダウンロードしてアプリをインストールしたり、変更されたファイルを更新します。必要なファイルをすべて処理した後、アプリが実行されます。



> Webサーバーと配布サーバーが物理的に別のサーバーを使用する必要はありません。各サーバーへのパスだけ明確に指定すると、機器の配置は必要によって調整することができます。

1.1.4 配布環境

配布とは、アーカイブファイルを配布サーバーに反映してnexacro platform アプリをユーザー端末にインストールすることを意味します。

アプリは、各端末のOSをサポートしているマーケットで配布します。 Androidは、Google、端末メーカー、通信事業者が運営する様々なマーケットを通じてアプリケーションが配布され、専用サーバーを通じて配布することもできます。iOSはApp Storeを通じて配布することができます。企業内でのみ使用するアプリケーションは、iOS開発者プログラム中のエンタープライズプログラム(Enterprise Program)を活用することができます。

Android 及び iOSで配布可能な方法は、次の通りです。

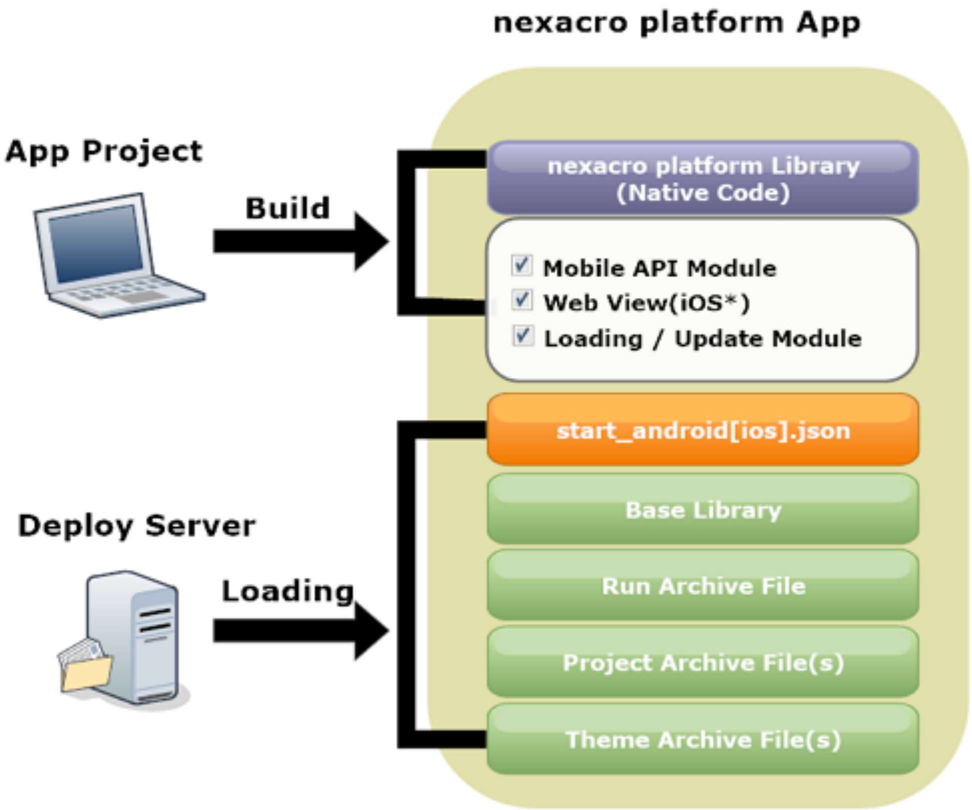
OS	配布方法	マーケット種類	内容	運営者
Android	マーケット	Google Play	Androidを開発した Googleのマーケットとして、最大数の Androidアプリケーションが配布されています。	Google
		Samsung Apps	メーカーであるサムスン電子が運営するアプリ配信マーケットです。	サムスン
	自体配布		私設ウェブサーバーに apkファイルを位置させて、該当 URLを使用して配布する方法です。	自体運営
iOS	マーケット	Apple Appstore	Appleが運営しているアプリ配信マーケットです。但し、私設サーバーを利用することはできません。	Apple
	独自配布		エンタープライズプログラムに登録して使用することができます。 https://developer.apple.com/programs/enterprise/	独自運営

アンドロイドOSは、サービスを提供する国によって使用可能なマーケットが変わる場合があります。上記の表には、韓国で利用できる主なマーケットを明示しました。

1.2 開発プロセス

1.2.1 主要構成

nexacro platformアプリの主要構成は、次の通りです。



nexacro platformアプリ(APK/APP)の主要構成についた内容は、次の通りです。

構成要素	詳細内容	含み時点
nexacro platform ライブラリ (ネイティブコード)	nexacro platform ライブラリはエンジンライブラリでアプリ開発時に必ず含める必要があるファイルです。iOS(nexacro14.framework), アンドロイド（nexacro14.jar, libnexacro14.so）形式で提供されます。 詳細設定は、次のとおりです。 • Mobile API モジュール Mobile API 動作のための Native Codeモジュール • WebView (iOS) 画面などを表現するためのWeb View Control • Loading/Update Module アプリの実行時にロード及びアップデートを処理するモジュール **※ 注意事項** • iOSオペレーティングシステムで使用するnexacro14.frameworkは、圧縮されたファイルで提供され、解凍してフォルダ状態で使用し、デバイスでテストする場合とシミュレータでテストするとき、それぞれ別のファイルを使用します。	Android Studio または、 Xcodeで、 nexacro platform アプリケーションプロジェクトに含ませてビルドします。
start_android.json start_ios.json	次のファイルのパス情報が含まれており、アプリケーションのビルド時に指定されたパスにアクセスして読み込みます。 • Runアーカイブファイル(iOS) *プロジェクトのアーカイブファイル *テーマのアーカイブファイル • nexacro platformライブラリ **※ 注意事項** • ファイル名は変更できますが、Android StudioやXcodeから指定した値と同じである必要があります。	実行時にロード及び更新で、nexacro platform アプリ内にダウンロードします。
Baseライブラリ	nexacro platformフレームワークを圧縮して提供します。JavaScriptで実装されたコンポーネントが含まれているnexacro14lib.zipファイルを使用します。 **※ 注意事項** • ファイル名は変更できますが、Android StudioやXcodeで指定した値と同じである必要があります。 • nexacro14libフォルダ構造をそのまま圧縮してファイルにする必要があります。	
Run	nexacro platformで使用するコンポーネントの情報と最初に実行するADLのJ	

構成要素	詳細内容	含み時点
アーカイブファイル	avaScriptファイルのパスが含まれているファイルです。 **※ 注意事項** iOSオペレーティングシステムを使用する場合にのみ必要です。 • 提供されるファイル名(Run.zip、Run.html)は、nexacro platform内で指定された値であるため、任意に変更しないでください。もし、ファイル名を変更すると動作しなくなります。	
プロジェクトアーカイブファイル / プロジェクトファイル	nexacro studioで開発したnexacro platform HTML5 プロジェクトファイルです。 **※ 注意事項** • 配備サーバーからプロジェクトのアーカイブファイルを一緒に配布することをお勧めします。アーカイブファイルは、Archive00.xzip形式で配布されます。 • 設定によって、アーカイブファイルを配布せずに直接HTTPサーバー上のプロジェクトにアクセスすることができます。作動方式は、配置されるアプリの特性に合わせて選択することができます。	
テーマアーカイブファイル / テーマファイル	nexacro platform HTML5プロジェクトのテーマファイルです。 **※ 注意事項** • 提供されるファイル名([テーマ名].zip)は、nexacro platform内で指定された値であるため、任意に変更しないでください。もし、ファイル名を変更すると動作しなくなります。 • 設定によって、アーカイブファイルを配布せずに直接HTTPサーバー上のプロジェクトにアクセスすることができます。動作方法は、配置されたアプリの特性に合わせて選択することができます。	

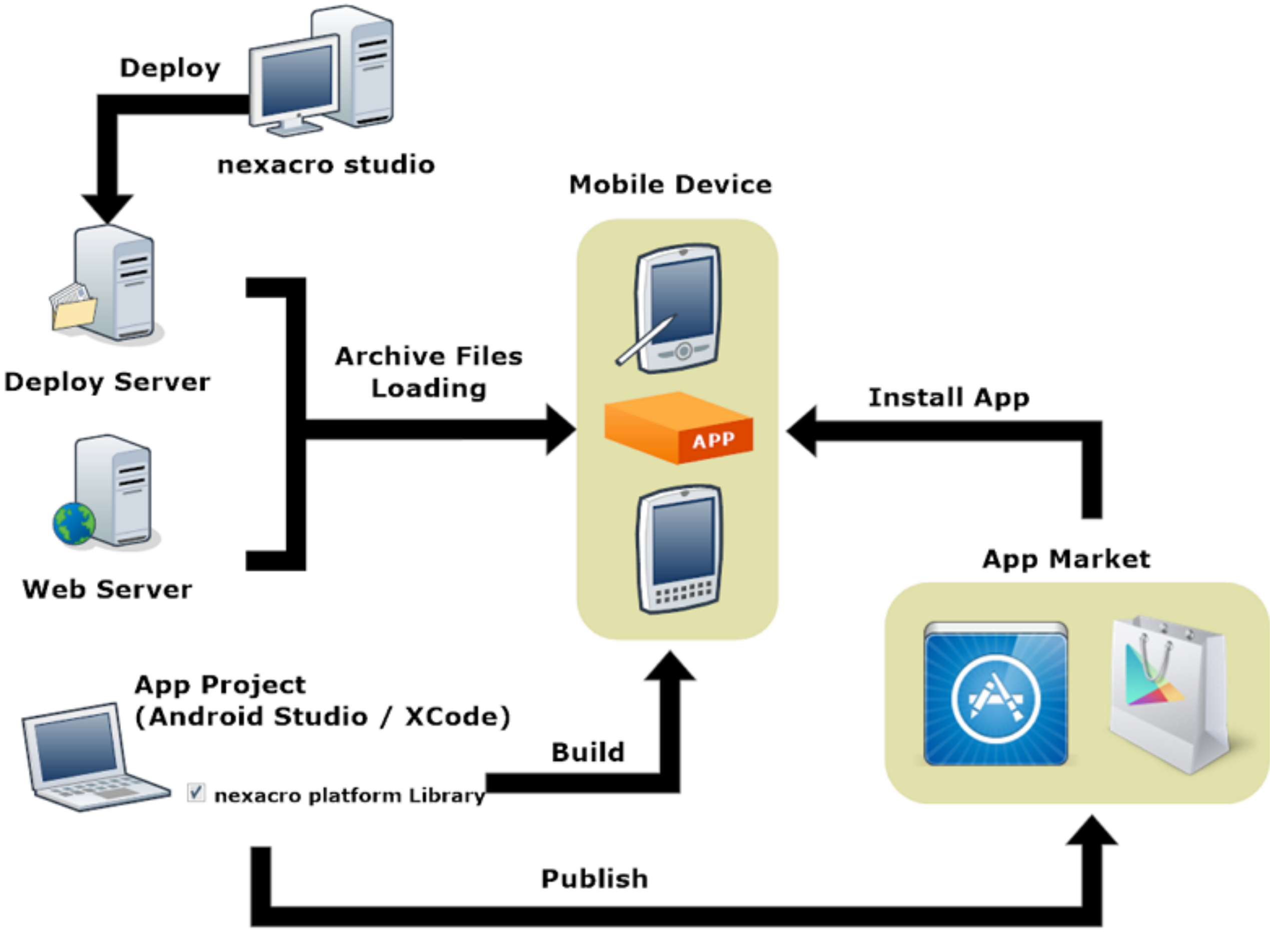
1.2.2 開発及び実行手順

開発及び実行には、次のような手順で進行します。

ステップ	詳細内容
プロジェクト開発	nexacro studioでnexacro platformアプリケーションを開発します。
配布サーバー反映	nexacro studioで必要なアーカイブファイルを生成した後、配布サーバーに反映します。
アプリ プロジェクトの作成とビルド	Android Studio(Android), Xcode(iOS)でアプリプロジェクトを生成します。プロジェクトを生成して提供されたnexacro platformライブラリを参照するようにします。 プロジェクトの設定が終わったらAndroid Studio(Android), Xcode(iOS)でアプリ(APK/APP)をビルドします。 ※ iOSの場合、端末をターゲットにビルドするためには、Appleの開発者アカウントが必要です。
実行	端末の接続後、Android Studio(Android)、Xcode(iOS)で端末に実行します。 ※ iOSの場合、端末をターゲットにビルドするためには、Appleの開発者アカウントが必要です。

1.2.3　配布手順と方法

nexacro platformは、下の図のように、アーカイブファイルとアプリが別々に配布されます。アーカイブファイルは、nexacro studioでパッケージングしたBaseライブラリ、プロジェクトアーカイブ、テーマアーカイブ、Runアーカイブなどで構成され、配布サーバを経由して渡されます。アプリはnexacro platformライブラリが含めて生成されたプロジェクトで作成され配布可能なファイルで作られてマーケットやプライベートサーバー（Androidのみ）に配布します。ユーザーがスマートフォン端末を駆動した後、マーケットやプライベートサーバでアプリをダウンロードして実行すると、アプリケーションが配布サーバから必要な情報をロードして画面を実行させせます。



マーケットを利用したnexacro platform Hybridアプリ(APK/ APP)配布手順は次のとおりです。詳細事項はマーケットごとに差があり、各マーケットの関連マニュアルを参照してください。

1. 開発者アカウントの生成
2. 配布用アプリ(APK/APP)生成
3. 検査登録

4. 検査完了
5. サービス開始

1.3　アプリ開発及び実行のための推奨仕様

Android、iOSのSDKのバージョン及び端末推奨仕様は以下のとおりです。

> 配布のためには、最新バージョンのツール及びSDKを利用することをお勧めします。

1.3.1　アンドロイド(Android)

区分	項目	内容
開発	OS	Window/Mac/Linux使用可能
	Android Studio	Android Studioのインストールファイルは、基本的に SDKツールが含まれています。 http://developer.android.com/sdk/index.html
	開発者 アカウント	Google Playへの配布のためには、各社の開発者アカウントが必要です。開発者アカウントの費用ポリシーは、各社ごとに異なります。
端末	端末OS	Android OS 2.3 以上 **※ 但し、上記のOS以外のは追加検討が必要**

1.3.2　iOS

区分	項目	内容
開発	OS	Mac OS Xでのみ開発/配布が可能 最小：Mac 10.6.8 以上
	Xcode	iOS用アプリの開発のためのツール(無料) https://developer.apple.com/xcode/
	iOS SDK	Xcodeのインストール時に一緒にインストールされます。(無料) Appstoreに配布するためには、最新バージョンのSDKでビルドされる必要があります。 https://developer.apple.com/ios/download/
	開発者 アカウント	Apple iOS開発者向けページから生成し、年間の費用形で販売されています。 価格は年間約1万円程度であり、為替レートによって違います。 https://developer.apple.com/jp/programs/ios/
端末	端末OS	iOS 9以上 **※ 但し、上記OS以外のことは、追加検討が必要**

第2章

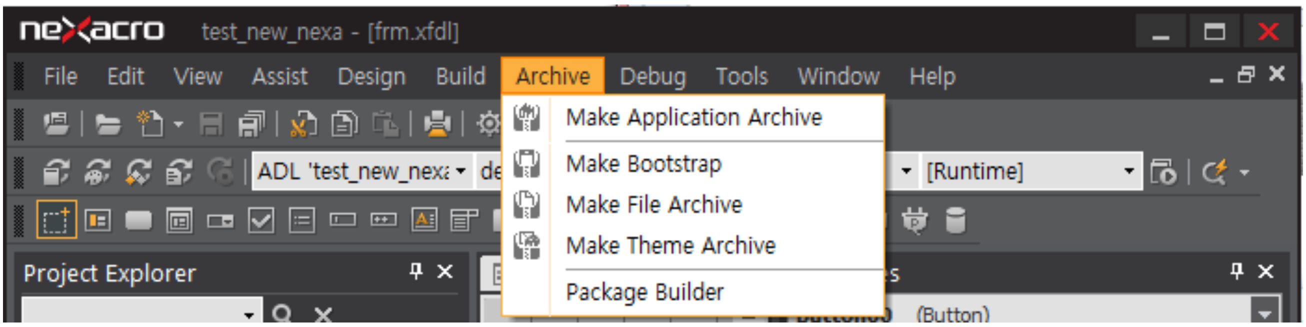
パッケージング及び配布

パッケージングとは、端末から実行時に必要な情報を作成及び圧縮する一連の過程を意味します。プロジェクト情報を利用したstart_android[ios].jsonの作成、bootstrapアーカイブファイル、ファイルアーカイブファイル、テーマアーカイブファイルを生成し、圧縮する過程を意味します。

パッケージングされたファイルは、配布サーバーにアップロードしてデバイスにインストール時にダウンロードして反映することになります。

パッケージングは、nexacro studioのArchiveメニューから実行します。 nexacro studioでは、配置するファイルを指定されたパスに生成し、サーバーに配布することは、作業環境によって適用する必要があります。

> パッケージングは、nexacro studioを使用する必要があり、エラーが発生する可能性がありますので任意で他のエディタで作成または変更しないでください。

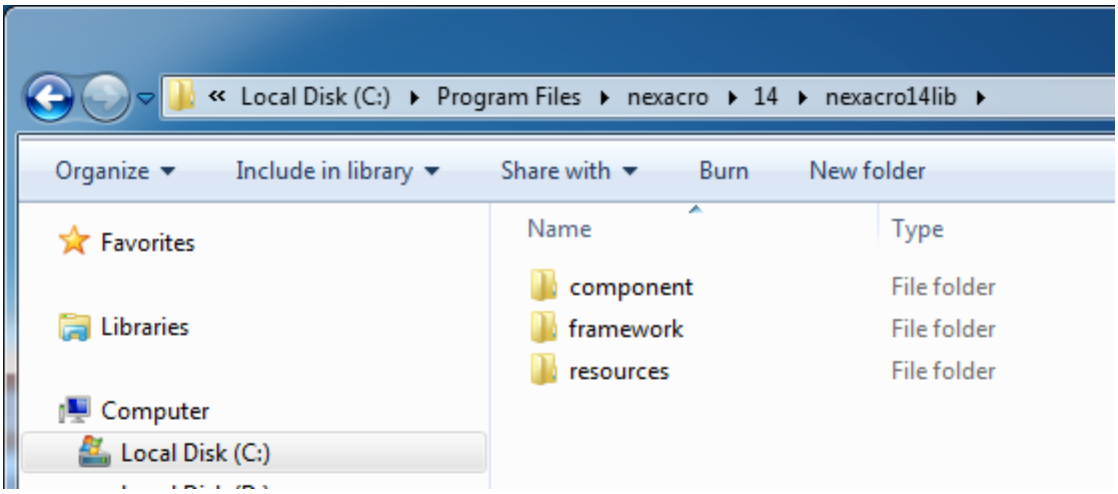


パッケージの段階で生成されるファイルは、以下のように区別することができます。ただし、Baseライブラリは直接生成する必要があります。

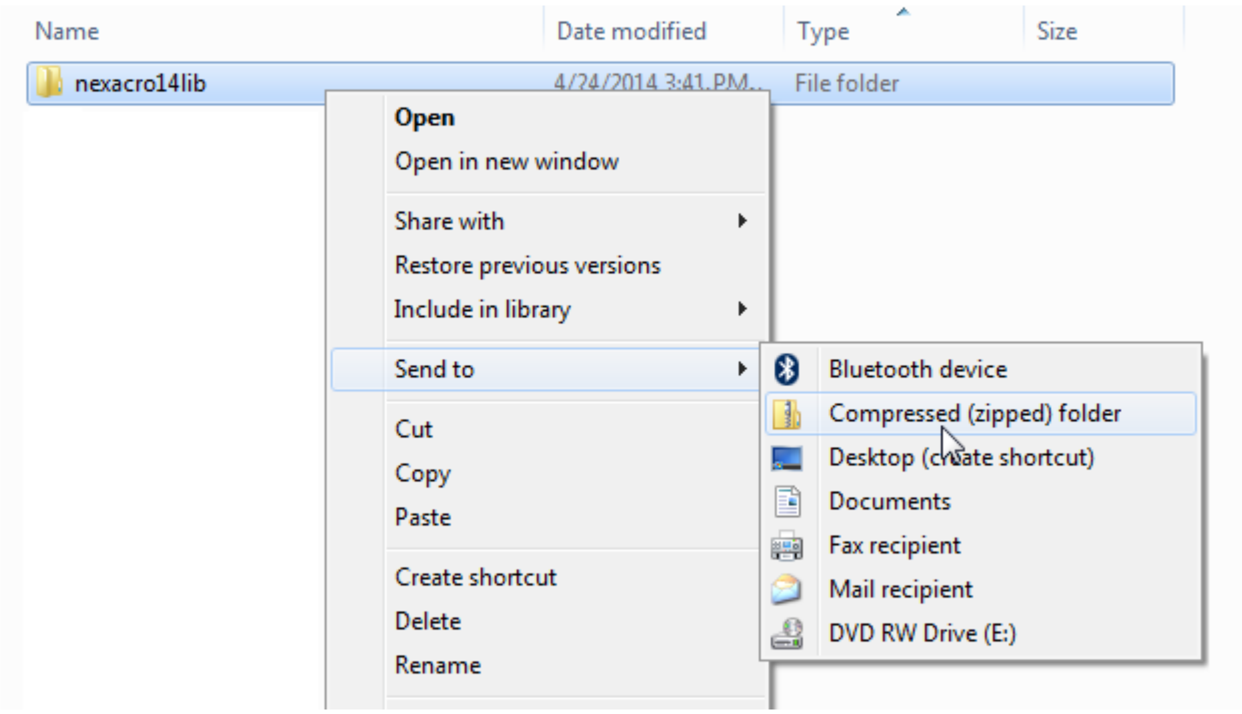
生成ファイル	Application	Bootstrap	File	Theme
テーマファイル	Y			Y
実行ファイル	Y	Y		
フォームファイル	Y		Y	
Baseライブラリ				

2.1　Baseライブラリ

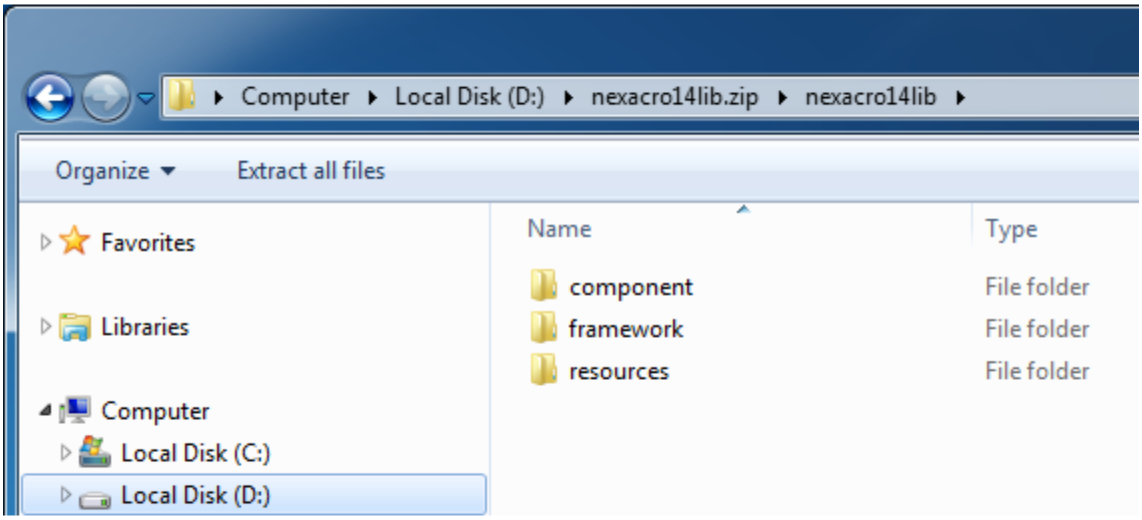
nexacro platformがインストールされたパスにアクセスすると、以下のようなフォルダを確認することができます。該当フォルダは、アプリケーションをWebでサービスする場合は、該当サーバーにそのままコピーして参照し、iOSやAndroid端末で使用する場合は、圧縮されたファイル形式で作成して使用します。



圧縮されたファイルに作成するときは、nexacro14libフォルダを別の場所にコピーして、該当フォルダをそのまま圧縮します。OSから基本的に提供される機能を使用するか、別のソフトウェアを使用することができます。



このとき、圧縮ファイルには、nexacro14libフォルダそのまま含まれている必要があります。



使用するソフトウェアによってnexacro14libフォルダを選択して圧縮するとサブフォルダであるcomponent、framework、resourcesのみ圧縮される場合があります。こんな場合には、上位に任意のフォルダを生成し、圧縮ファイルを生成すると、nexacro14libフォルダそのまま圧縮することができます。

Run Templateファイルの生成時に'Merge JSON File'オプションを選択する場合には、[Build> Deploy> Deploy Application]メニューから'Compress JS Files'、' Merge JSON File'オプションを選択して、生成されたnexacro14libフォルダを圧縮して使用する必要があります。

2.2 Make Application Archive

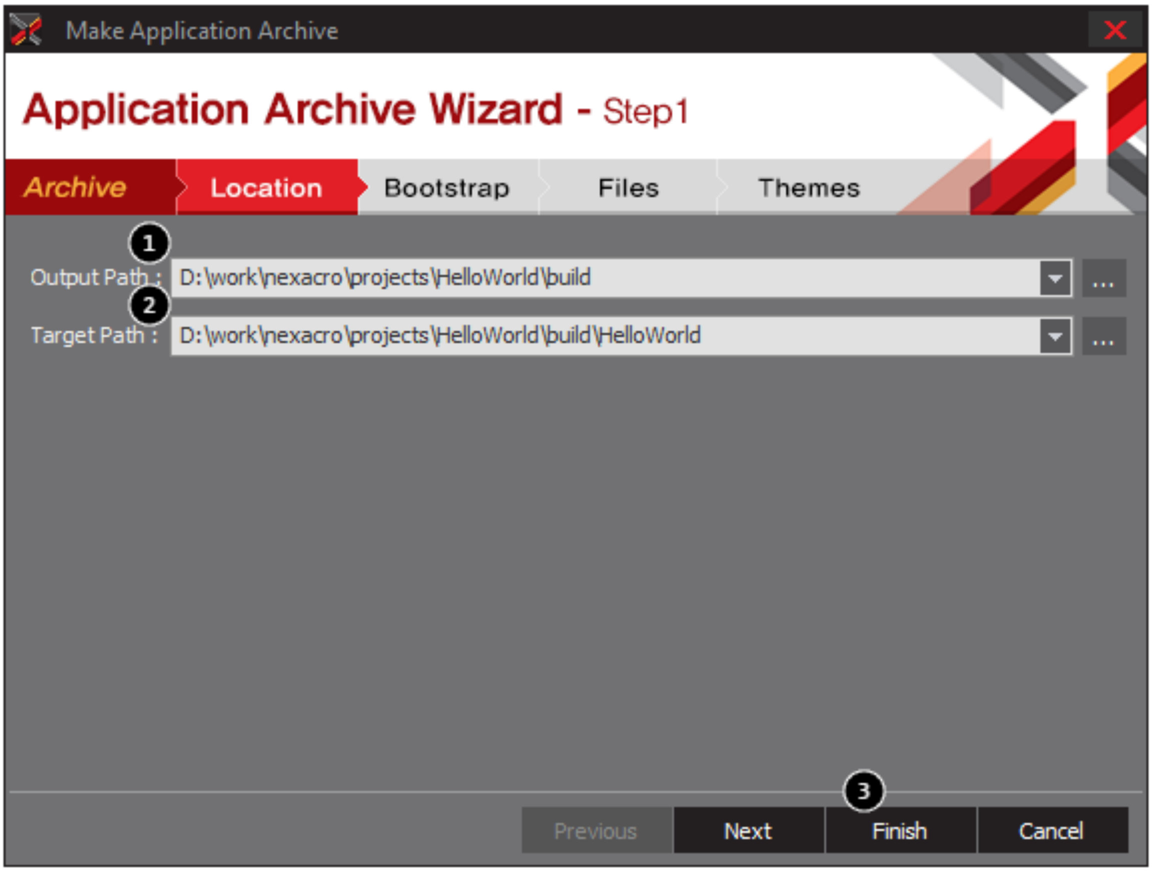
Make Bootstrap、Makefile Archive、Make Theme Archive機能を一度に処理することができる機能を提供します。

1. [nexacro studio > Archive > Make Application Archive]メニューをクリックします。

2. パス情報、Bootstrap、プロジェクトファイル、テーマ情報を次の順序とおり設定します。

Step 1

アーカイブファイルを生成するパスとプロジェクトのパスを指定します。

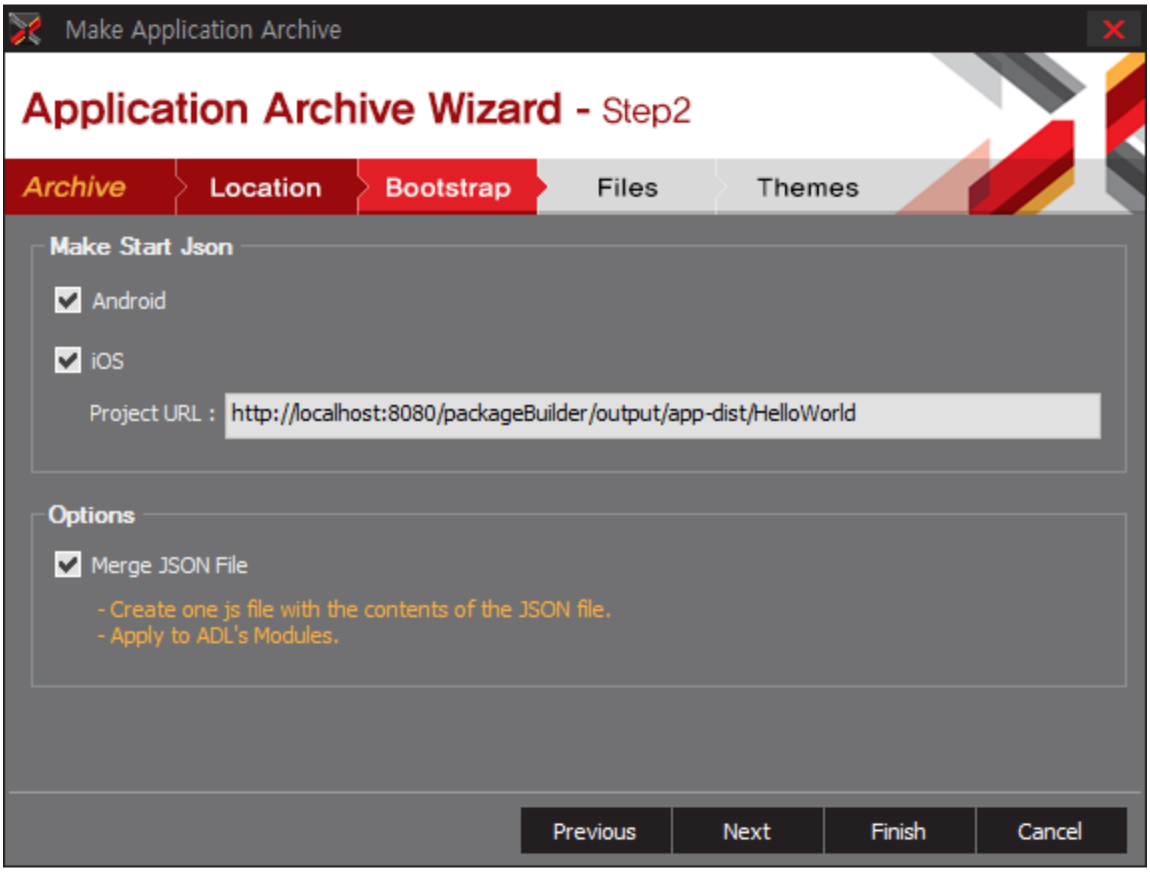


	項目	設定
❶	Output Path	アーカイブファイルを生成するパスを指定します。
❷	Target Path	プロジェクトがコンパイルされたパスを指定します。 プロジェクトのコンパイルパスを保存しておいたら、自動的に現在のプロジェクトのコンパイルパスが入力されます。
❸	Finish	デフォルト設定でアーカイブファイルを作成します。 • Run.zipと start_android[ios].jsonファイル(Merge JSON Fileオプションは以前の設定値を適用) • プロジェクトアーカイブファイル(Output Pathに設定したアーカイブファイルのリストを確認した後再生成) • テーマアーカイブファイル(現在活性化になっている ADLで使用しているテーマが適用される)

Step 2

Bootstrap関連オプションを設定します。

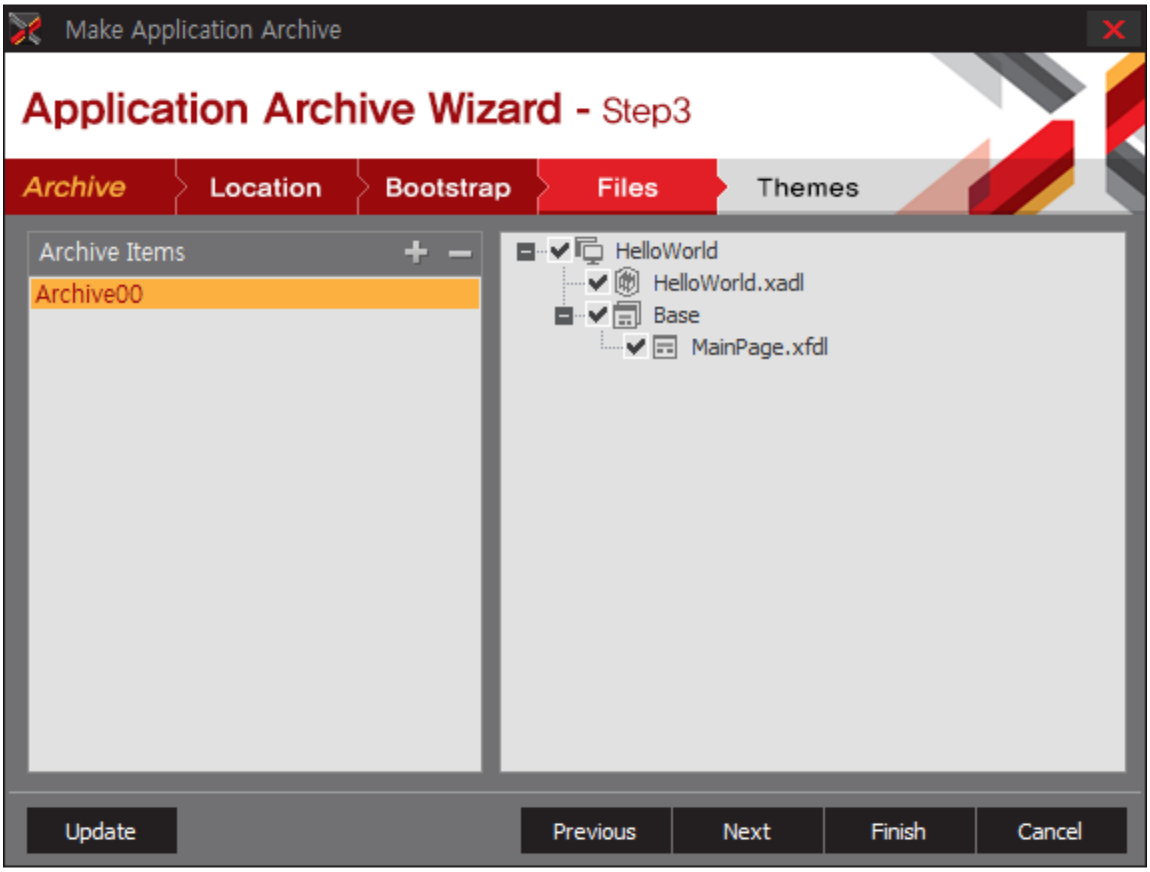
設定の詳細については、Make Bootstrapを参照してください。



Step 3

アーカイブファイルを生成するための設定を行います。

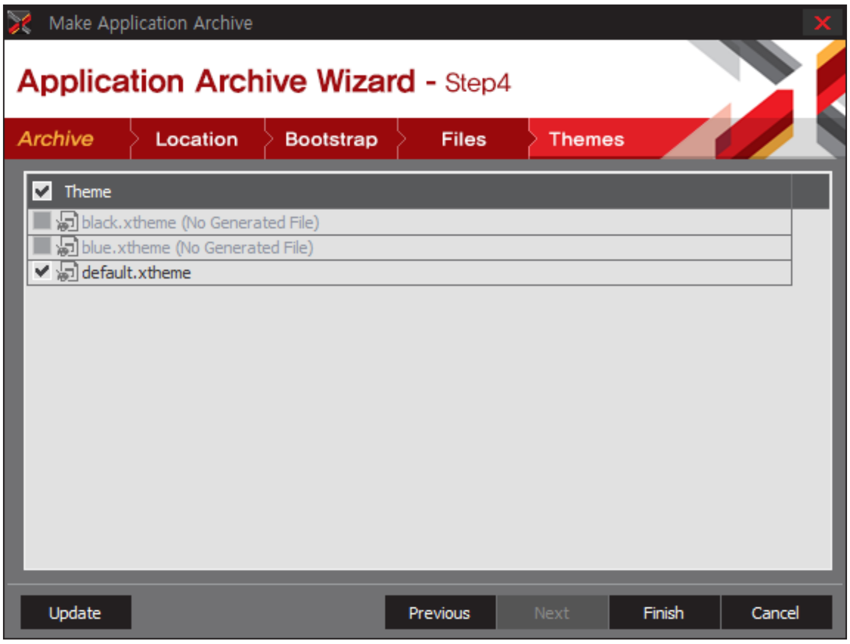
設定の詳細については、Make File Archiveを参照してください。



Step 4

テーマファイルをアーカイブファイルで作成するためにテーマを設定します。

> 設定の詳細については、Make Theme Archiveを参照してください。



3. 生成されたファイルを配布サーバーに反映します。

2.3 Make Bootstrap

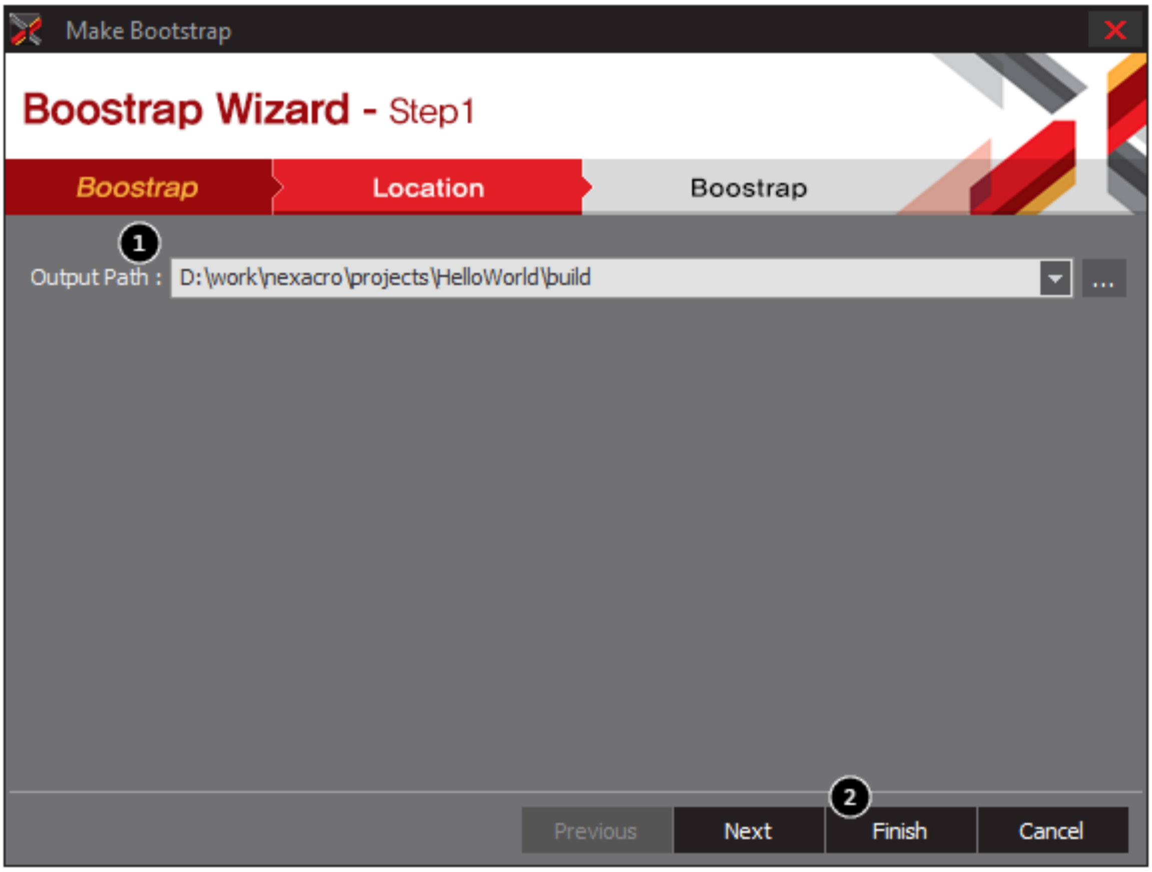
Appを駆動させるRun.htmlを含むRun.zip及びstart_android[ios].jsonファイルを作成する機能です。

1. [nexacro studio > Archive > Make Bootstrap]メニューをクリックします。

2. Bootstrapアーカイブファイルである Run.zipとstart_android[ios].jsonファイルを作成します。 start_android[ios].jsonファイルは、[Archive> Make File Archive]のステップで入力した情報を基に生成されます。

Step 1

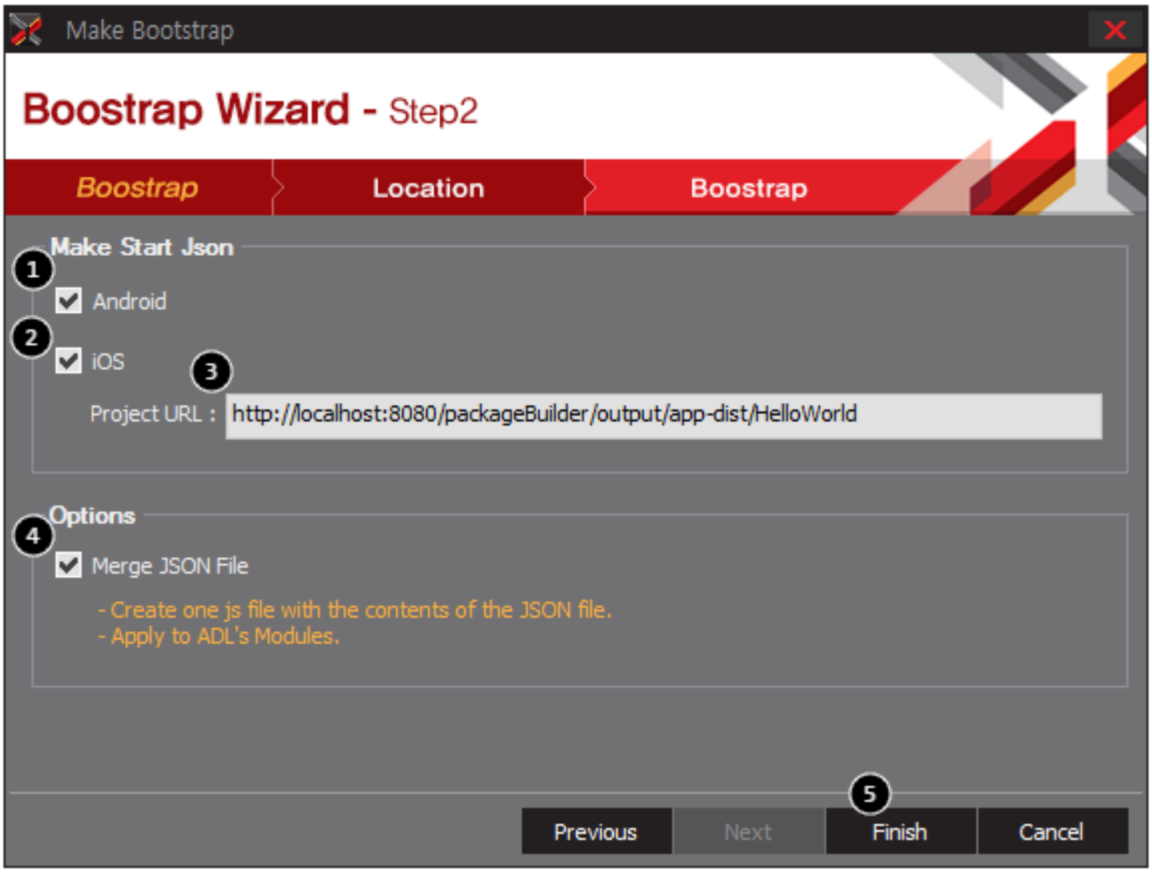
Run.zip, start_android[ios].jsonファイルを生成するパスを指定します。



	項目	説明
❶	Output Path	Run.zipと start_android[ios].jsonファイルが生成されるパスです。start_android[ios].jsonには、TypeDefinitionの Updateタブに設定された内容が含まれます。
❷	Finish	Run.zipと start_android[ios].jsonファイルを生成します。Merge JSON Fileオプションは、以前の設定値が適用されます。

Step 2

生成するアプリのモバイルプラットフォームを選択して、Project URLを指定します。 Android / iOSのいずれかは、必ず選択する必要があります。



	項目	説明
❶	Android	アンドロイドアプリを駆動させるために必要な start_android.jsonファイルを生成するかを選択します。
❷	iOS	iOSアプリを駆動させるために必要な start_ios.json、Run.zipファイルを生成するかを選択します。
❸	Project URL	最初のアプリケーションを実行するサーバーのURLを指定します。 ❷をチェックした場合にのみ、入力ウィンドウが活性化されます。
❹	Merge JSON File	nexacro platformライブラリを呼び出す方法を指定します。 該当オプションを選択した場合には、[Build > Deploy > Deploy Application]メニューから「Compress JS file」、「Merge JSON file」オプションを選択して、生成されたJavaScriptファイルを圧縮ファイルに作成して配布する必要があります。
❺	Finish	Run.zipと start_android[ios].jsonファイルを生成します。 ❶, ❷ の何一つチェックしていない場合には、start android[ios].jsonファイルを生成しません。

> TypeDefinitionでURLパスの設定時にローカルホスト(127.0.0.1)を指定すると、配布サーバーではなく、端末自身をローカルホストとして認識することになります。端末からアクセス可能なプライベートIPまたはURLを入力する必要があります。

3. 生成されたstart_android[ios].jsonファイルは、次のように生成されます。製品のアップデートによって、一部の内容が変更されることがあります。

   オフラインで使用できるようにプロジェクトのアーカイブファイルとテーマのアーカイブファイルを作成する場合は、該当ファイルを先に作成してからstart_android[ios].jsonファイルを作成する必要があります。

```
{
    "version" : "14.0.0.1",
    "runtime" :
    {
        "target" :
        {
            "os" : "android"
        },
        "framework" :
        [
         "./nexacro14lib/framework/Framework.js",
         "./nexacro14lib/framework/Device.js",
         "./nexacro14lib/framework/Device_Android.js"
        ],
        "component" :
        [
         "./nexacro14lib/component/CompBase.js",
         "./nexacro14lib/component/ComComp.js",
         "./nexacro14lib/component/Grid.js",
         "./nexacro14lib/component/DeviceAPI.js",
         "./archivefiles.js"
        ],
        "form" :
        [
        ],
        "xadl" :
        {
            "launch" : "./HelloWorld.xadl.js",
            "quickview" : "./HelloWorld.xadl.quickview.js"
```

```
        },
        "protocol":
        {
            "adaptors":
            [

            ]
        },
        "resource" :
        {
            "android" : {
                "updateurl" : "http://172.20.20.20:8080/packageBuilder/output/
app-dist/HelloWorld",
                "all" : {
                    "updateurl" : "",
                    "all" : {
                        "updateurl" : "",
                        "resources" :
                        [{"type" : "File", "file" : "Archive00.xzip", "
targetpath" : "", "version" : "1.0", "failpass" : "false"},
                         {"type" : "Engine", "file" : "nexacro14lib.zip", "
targetpath" : "", "version" : "1.0", "failpass" : ""},
                         {"type" : "Theme", "file" : "default.zip", "targetpath
" : "./_theme_", "version" : "1.0", "failpass" : "false"}]
                    }
                }
            },
            "ios" : {
                "updateurl" : "http://localhost:8080/packageBuilder/output/app-
dist/HelloWorld",
                "all" : {
                    "updateurl" : "",
                    "all" : {
                        "updateurl" : "",
                        "resources" :
                        [{"type" : "File", "file" : "Archive00.xzip", "
targetpath" : "", "version" : "1.0", "failpass" : "false"},
                         {"type" : "Engine", "file" : "nexacro14lib.zip", "
targetpath" : "", "version" : "1.0", "failpass" : ""},
                         {"type" : "Engine", "file" : "Run.zip", "targetpath" :
"", "version" : "1.0", "failpass" : ""},
```

```
                                                    {"type" : "Theme", "file" : "default.zip", "targetpath
" : "./_theme_", "version" : "1.0", "failpass" : "false"}]
                                          }
                                  }
                          }
                  }
          }
  }
```

Android OSではRun.zipファイルを使用しておりません。
Android, iOSそれぞれ必要なファイルを生成します。
- start_android.json
- start_ios.json
各ファイルのversion値を変更すると、アプリの実行時にファイルを更新します。

4. 生成されたファイルを配布サーバーに反映します。

2.4 Make File Archive

プロジェクトファイルをアーカイブファイルに作成し、TypeDefinitionで配布に必要な情報を登録する機能です。

アーカイブファイルは、ユーザーが指定した名前で作成され、xzip拡張子で保存します。

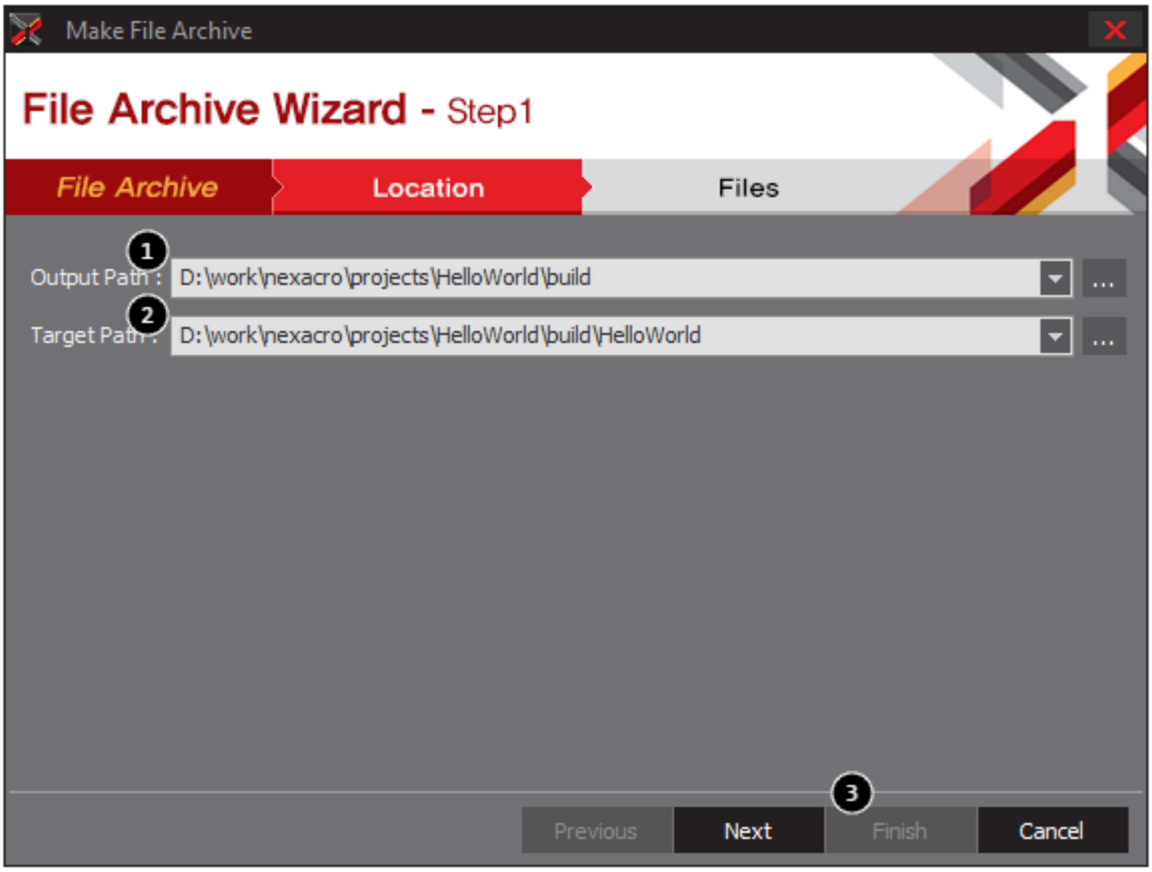
ファイルアーカイブファイルは、アプリが最初にインストールされる時、配布サーバーに接続してダウンロードし、以降には端末に保存されたデータで実行します。

1. [nexacro studio > Archive > Make File Archive]メニューをクリックします。

2. アーカイブファイル内に含まれるファイル内に含むADL及び画面などを選択します。

Step 1

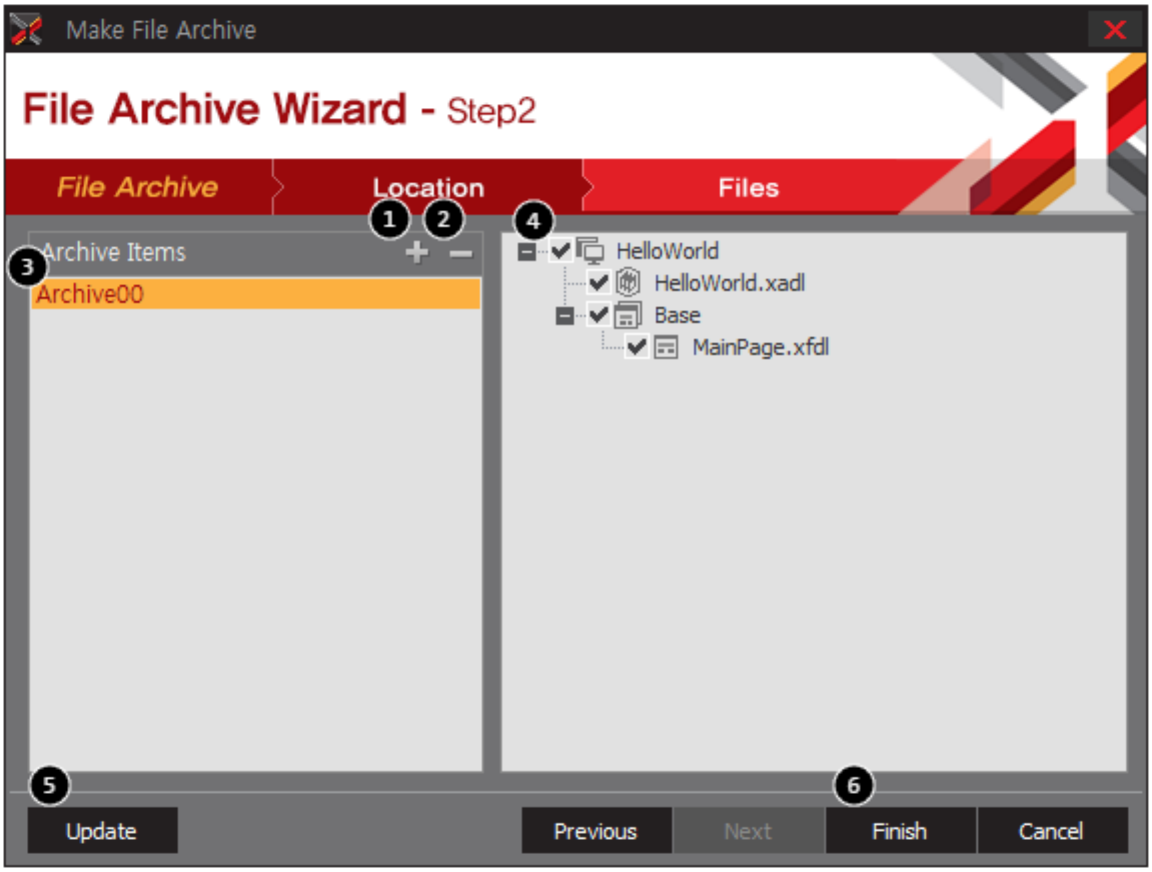
アーカイブファイルを生成するパスを指定します。



	項目	説明
❶	Output Path	アーカイブファイルを生成するパスを指定します。
❷	Target Path	プロジェクトがコンパイルされたパスを指定します。 プロジェクトのコンパイルパスを保存しておいたら自動的に現在のプロジェクトのコンパイルパスが入力されます。
❸	Finish	プロジェクトファイルをアーカイブファイルに生成します。

Step 2

アーカイブファイルとそのファイルに入れる ADL、フォーム、スタイルシートなどのリストを構成します。

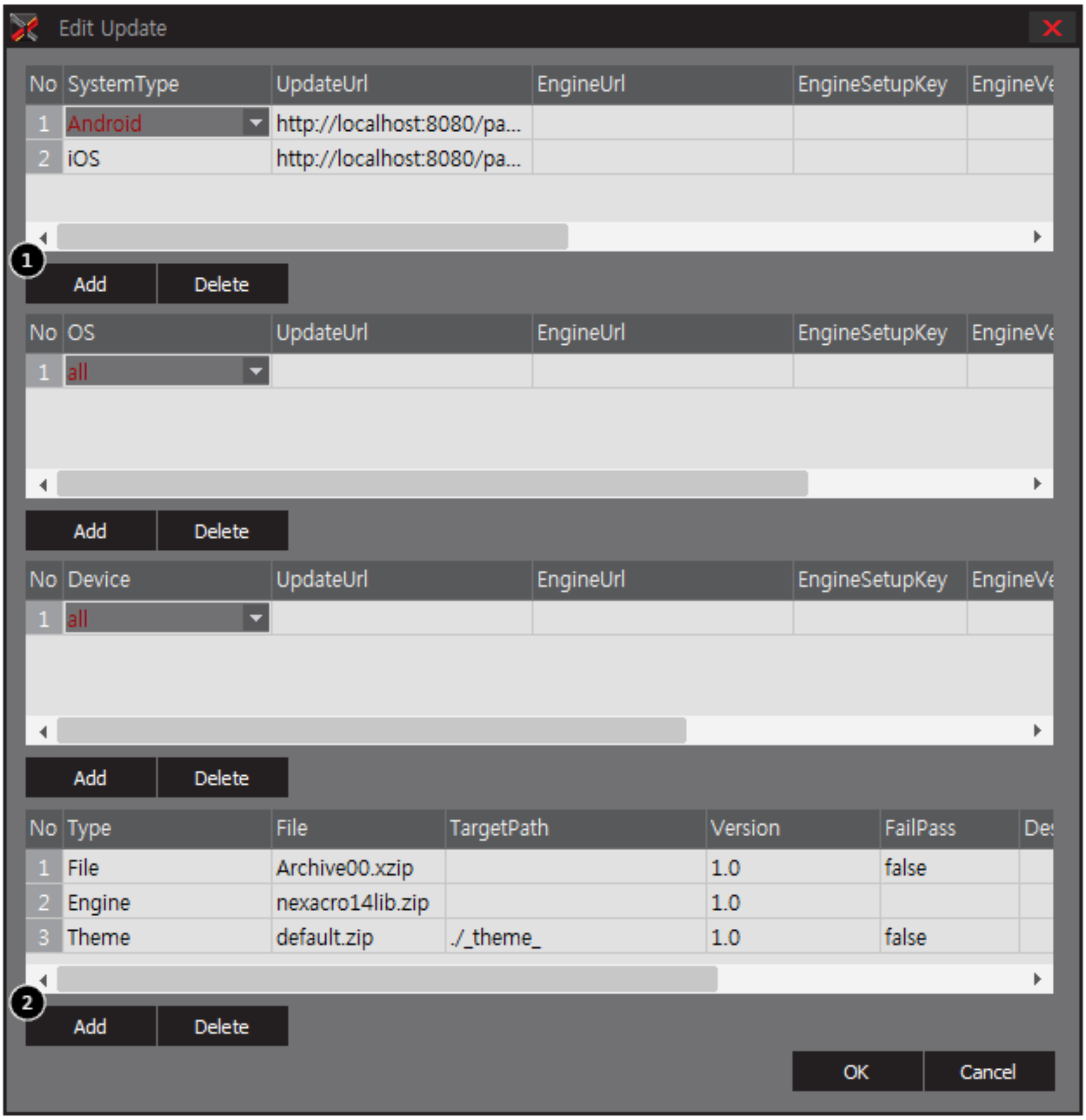


	項目	説明
❶	Add	アーカイブを追加します。
❷	Delete	選択されたアーカイブを削除します。
❸	Archive	生成するアーカイブファイルのリストです。 少なくとも1つ以上のアーカイブを追加する必要があります。
❹	Select File	アーカイブの生成時に構成するプロジェクトファイルを選択することができます。

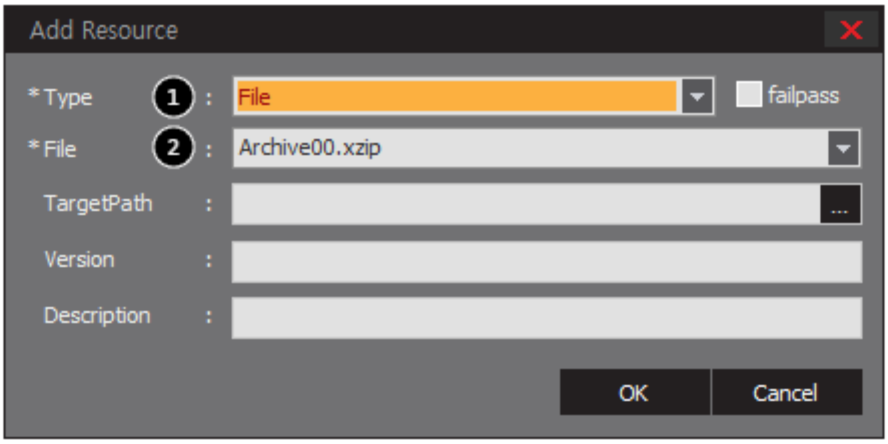
	項目	説明
		Step1で指定された Output Pathに、以前に作成したアーカイブファイルがある場合には、現在のプロジェクトファイルと比較して一致するとアーカイブファイルをロードしてツリー形式で表示します。 少なくとも1つ以上のファイルを選択する必要があります。
❺	Update	TypeDefinitionにUpdate(Resource)にアーカイブファイルを手動で追加することができます。
❻	Finish	TypeDefinition項目がすでに入力されているときに、該当手順をスキップしてすぐアーカイブファイルを生成します。

Step 3

TypeDefinitionに Update項目を登録します。



	項目	説明
❶	Add	SystemTypeを生成した後、OS、Device、Resourceを追加します。 Device項目に特定 Deviceごとに違う設定が必要なときに system.mobileproducttypeプロパティ値と一致する値を指定する必要があります。 例) SHW-M250S, Nexus One ※ 詳細項目として次の項目が設定可能です。 - UpdateURL：コンポーネントライブラリを更新するためのURLを指定するプロパティです。
❷	Add	Step2で生成されるアーカイブファイルを Resourceに登録します。



	項目	説明
❶	Type	登録可能なタイプは、File、Engine、Themeです。 プロジェクトアーカイブとテーマアーカイブを生成せずに Webサーバーから直接ダウンロードしようとする場合には、Engine項目にみ指定します。
❷	File	ファイル名は、上記のStep2で作成したアーカイブ名をリストから選択することができます。

3. 生成されたアーカイブファイルを配布サーバーに手動で反映されます。

2.5 Make Theme Archive

テーマファイルをアーカイブファイルに作成する機能です。

アーカイブファイルは、ユーザーが指定したテーマ名に生成され、生成されたファイル名は任意に変更できません。

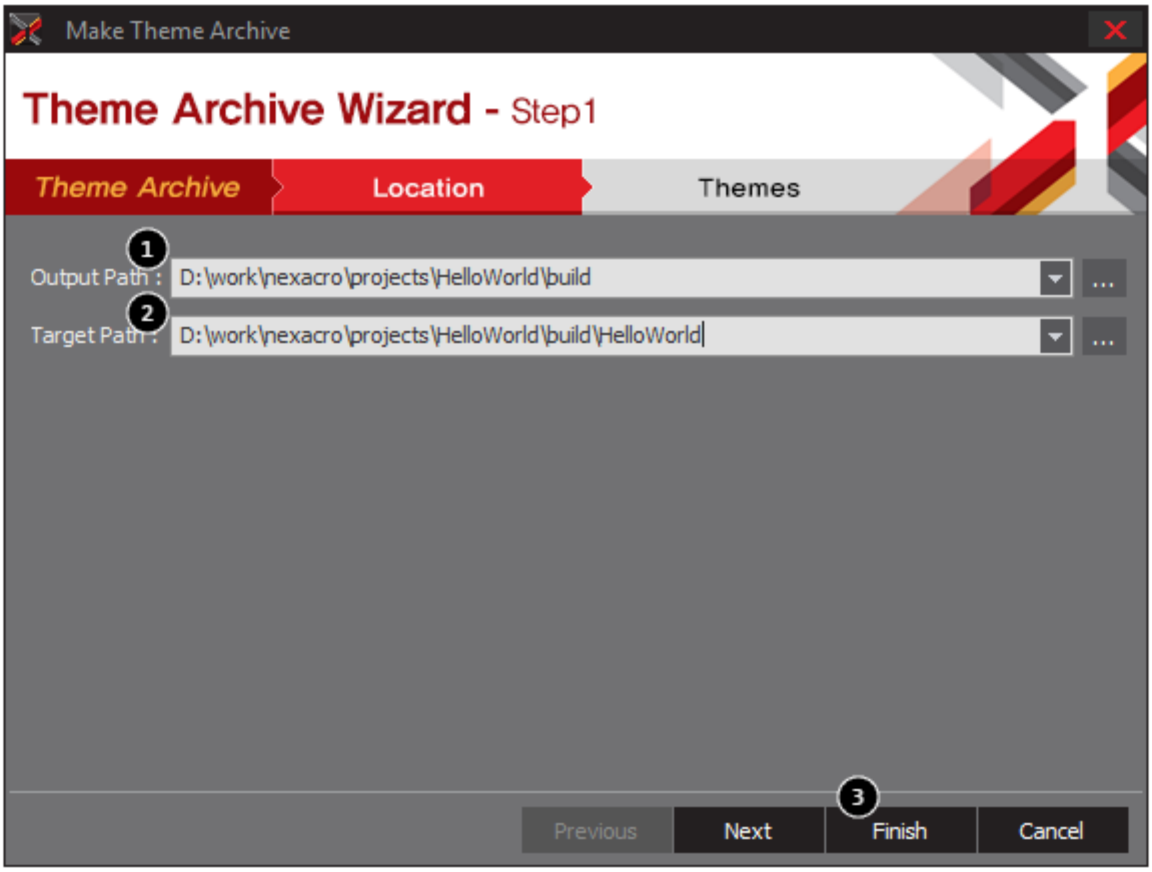
テーマアーカイブファイルは、アプリが最初にインストールされる時、配布サーバーに接続してダウンロードし、以降には端末に保存されたデータで実行します。

1. [nexacro studio > Archive > Make Theme Archive]メニューをクリックします。

2. JavaScriptファイルに変換されたテーマアーカイブを作成します。

Step 1

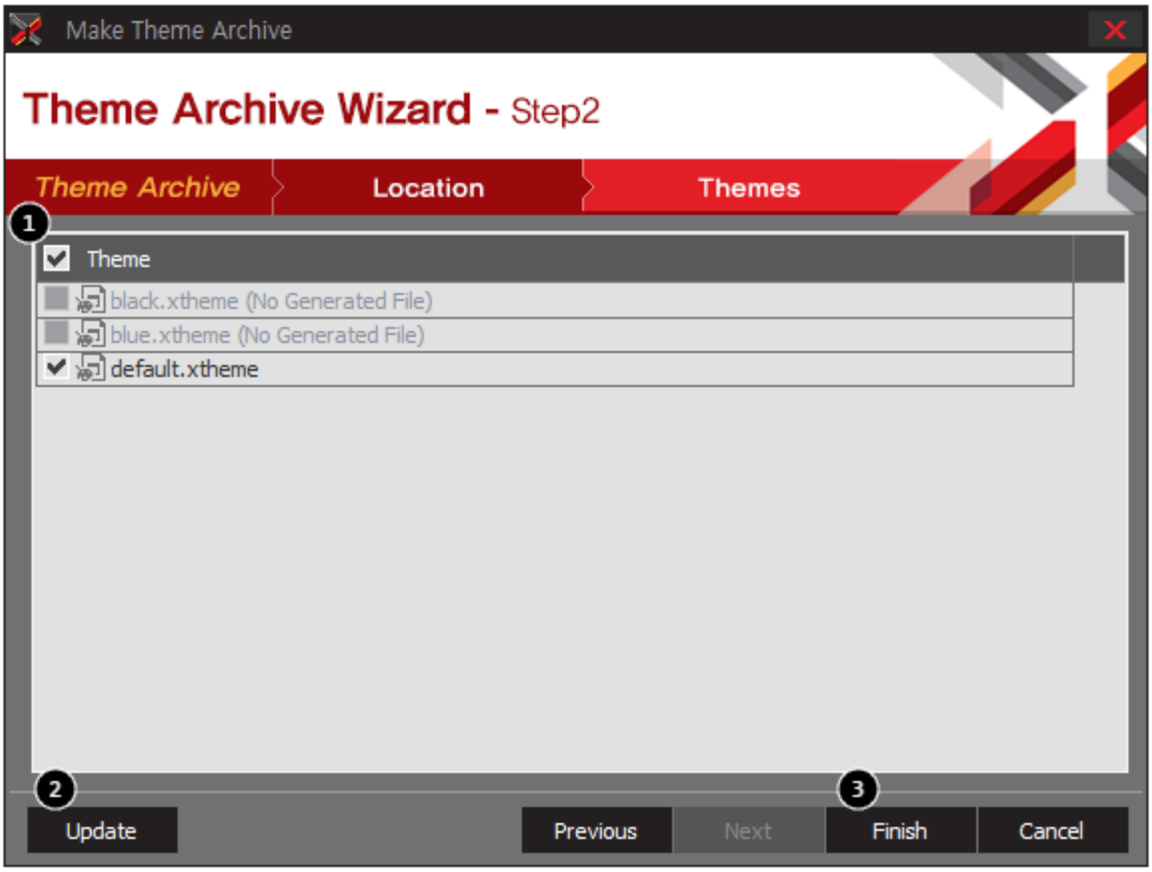
テーマアーカイブファイルを生成するパスを指定します。



	項目	説明
❶	Output Path	圧縮されたアーカイブファイルを生成するパスを選択します。
❷	Target Path	HTML5でコンパイルされたプロジェクトのパスを選択します。HTML5プロジェクトのコンパイル時にパスを保存しておいたら、自動的に現在のプロジェクトのコンパイルパスが入力されます。
❸	Finish	現在活性化になっている ADLで使用しているテーマが適用されたテーマアーカイブファイルを生成します。

Step 2

アーカイブファイルで生成するテーマを選択します。



	項目	説明
❶	Theme	アーカイブする対象となるテーマを選択します。 選択テーマは HTML5であらかじめコンパイルされている必要があり、コンパイルされたファイルは、[Compiled Path]/../_theme_ フォルダにある必要があります。 現在のプロジェクトで使用中の Activeテーマ、スクリーンカスタムテーマのリストを自動的にチェックして表示します。 少なくとも1つ以上のテーマを選択する必要があります。

	項目	説明
❷	Update	TypeDefinitionに Update(Resource)にアーカイブファイルを手動で追加することができます。
❸	Finish	選択したテーマを使ってアーカイブファイルを生成します。

3. 生成されたアーカイブファイルを配布サーバーに手動で反映します。

第3章

アプリ開発と実行(iOS)

3.1　事前準備

iOSアプリを開発して配布するためには下記のような事前準備が必要です。各ステップで必要な作業の詳細は、Appleが提供するガイドを参照してください。

- Appleのユーザーアカウント作成（Apple ID）
- 開発者プログラム登録（Apple Developer Program）
  企業用の開発者プログラムの登録の際には、DUNSナンバーが必要な場合があります。
- 証明書発行
  Certificate Signing Request（CSR）の発行およびcertificate（cer）ファイルを生成します。
- App ID登録
- Device登録
  iTunes、Xcodeでアプリをインストールし、テストする際に必要です。
- Provisioning Profile生成

nexacro studioで生成したアプリケーションをデバイス上で実行するためには、インストール可能なアプリにビルドする作業が必要です。ビルド作業のためにはXcode、iOS SDKが必要です。

> 管理者ガイドではXcode7.3.1バージョンを基準に説明します。
> Xcodeバージョンの更新により、一部メニューや動作が変更されることがあります。

3.2　XcodeプロジェクトでのiOSアプリ開発およびビルド

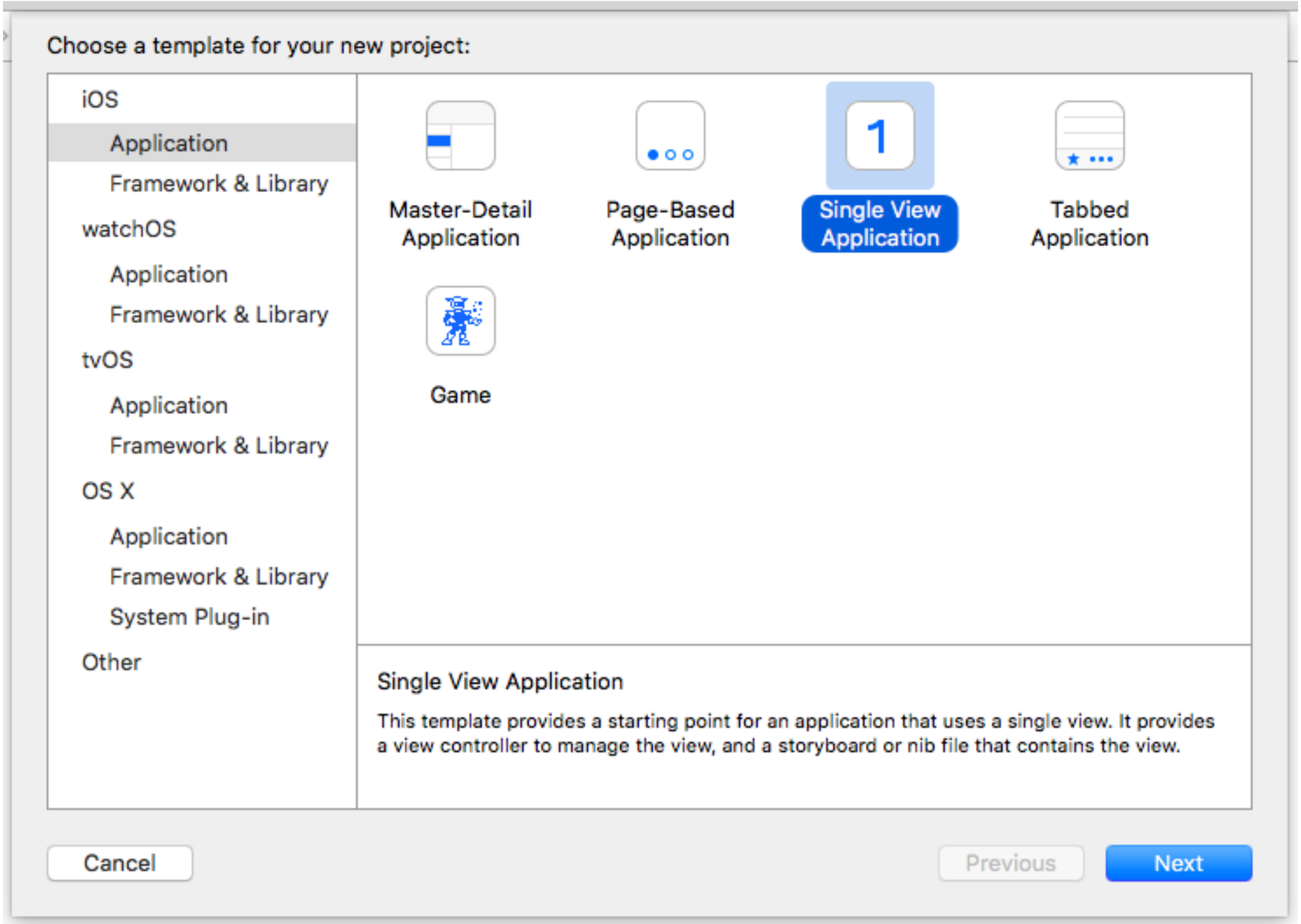
Xcodeで以下のような手順に従って、iOS版のアプリを開発します。Xcodeでのプロジェクトを進めるには、nexacro studioで作成したnexacroアプリケーションのアーカイブファイルが指定された場所に配置されている必要があります。

3.2.1　プロジェクト生成

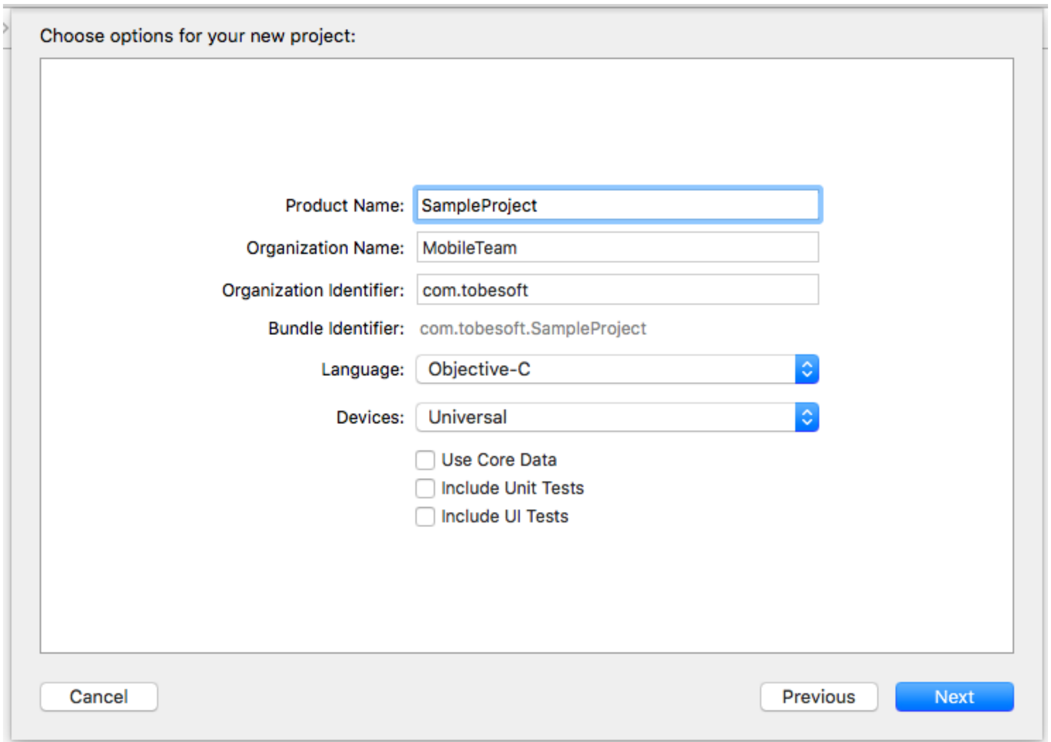
nexacro platformで開発されたアプリケーションをiOS版のRuntimeアプリとして実行するために、Xcodeプロジェクトを作成し、デフォルト環境を設定する必要があります。新しいプロジェクトは、以下のメニューから生成することができます。

```
File > New > Project
```

プロジェクトを生成するためのテンプレート画面から「Single View Application」を選択し、「Next」ボタンをクリックします。



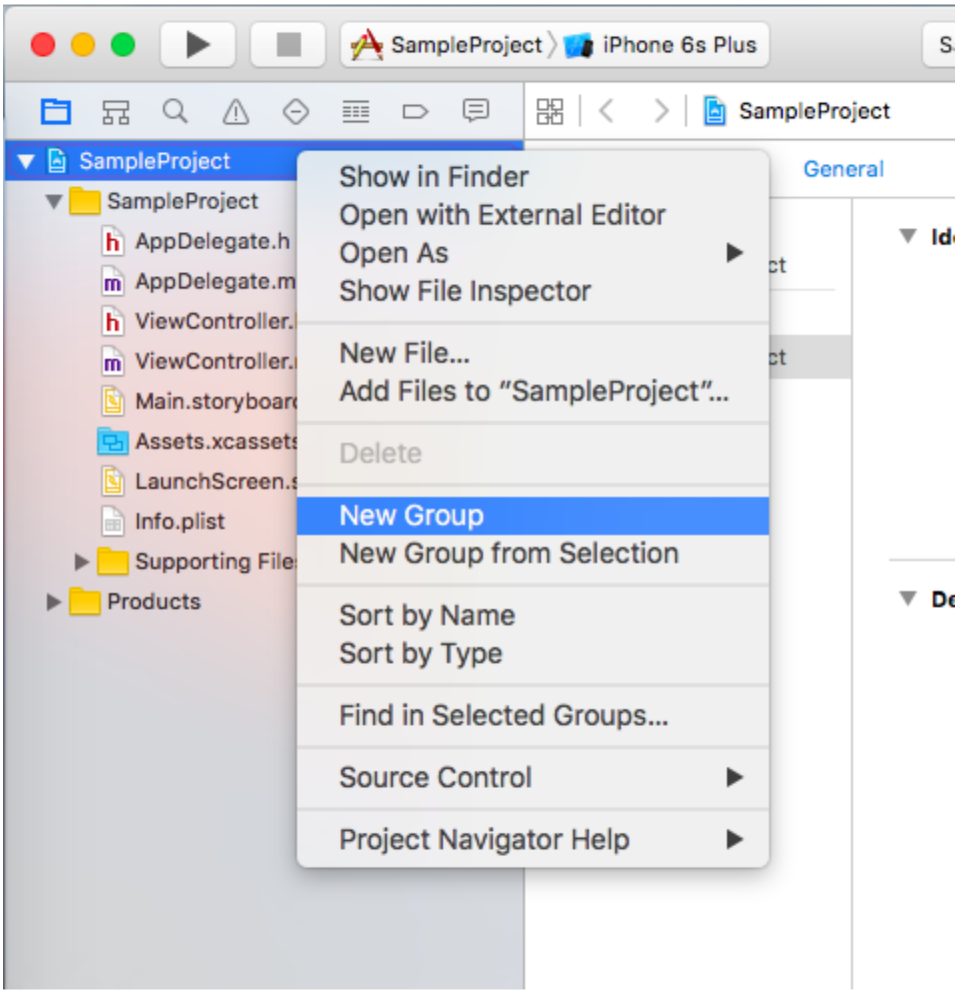
Product Nameやその他必要な項目を設定し、「Next」ボタンをクリックした後に表示される画面からプロジェクトを生成する場所を指定し、「Create」ボタンをクリックします。



iPhone Xにて、画面上下に余白を表示しないためには、プロジェクトにLaunchScreenのstoryboardを設定する必要があります。
詳細については、以下のリンクを参照してください。
https://developer.apple.com/ios/update-apps-for-iphone-x/
https://developer.apple.com/ios/human-interface-guidelines/icons-and-images/launch-screen/

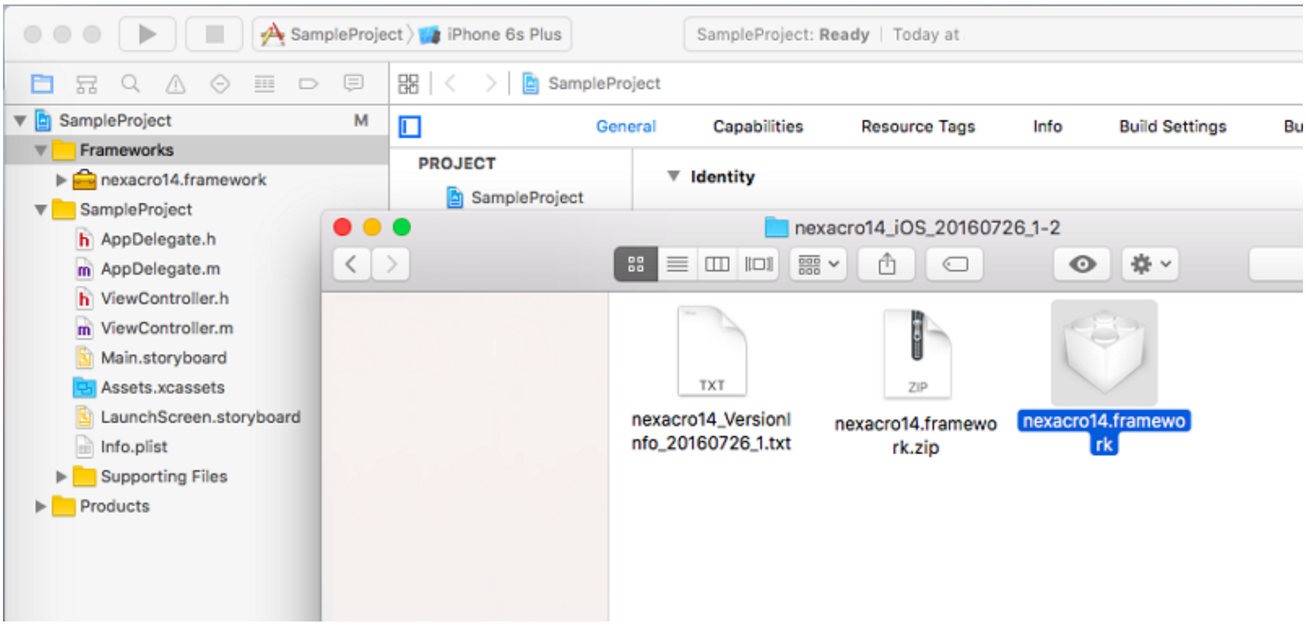
3.2.2 nexacro platformライブラリの設定

Xcodeプロジェクトで、nexacro platformに最適化された環境を作成するためのnexacro platformライブラリファイルを追加する必要があります。プロジェクトを選択し、「Frameworks」フォルダを追加します。



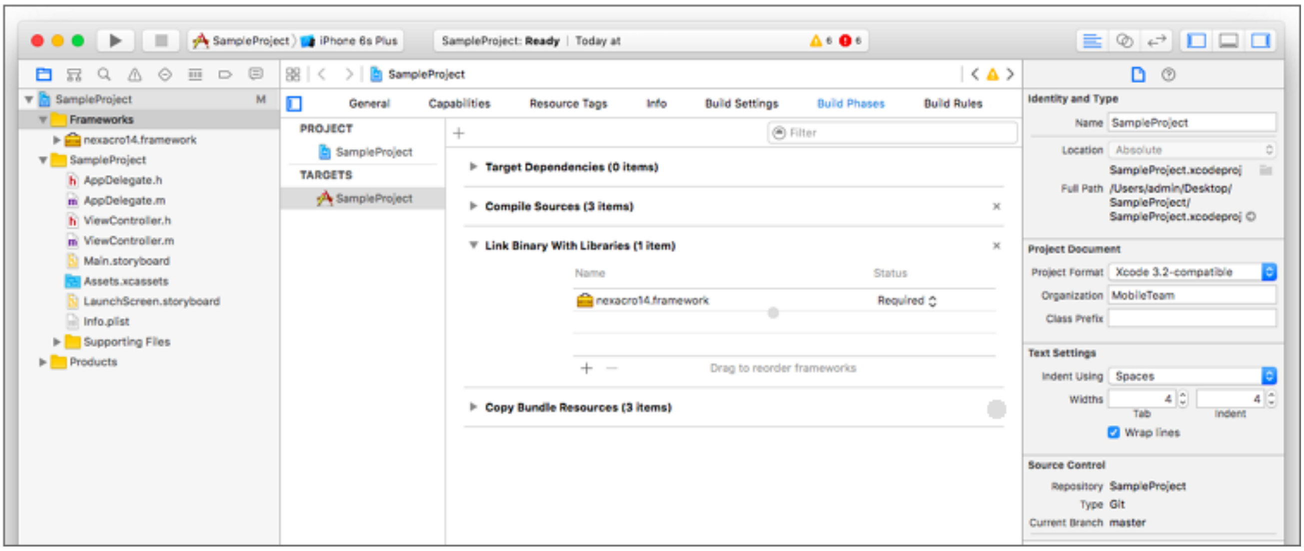
nexacro platformライブラリは、圧縮ファイル形式の「nexacro14.framework.zip」というファイル名で提供されます。圧縮ファイルを解凍してから、プロジェクトのFrameworksフォルダにドラッグ＆ドロップし

ます。



3.2.3　追加ライブラリ/フレームワーク設定

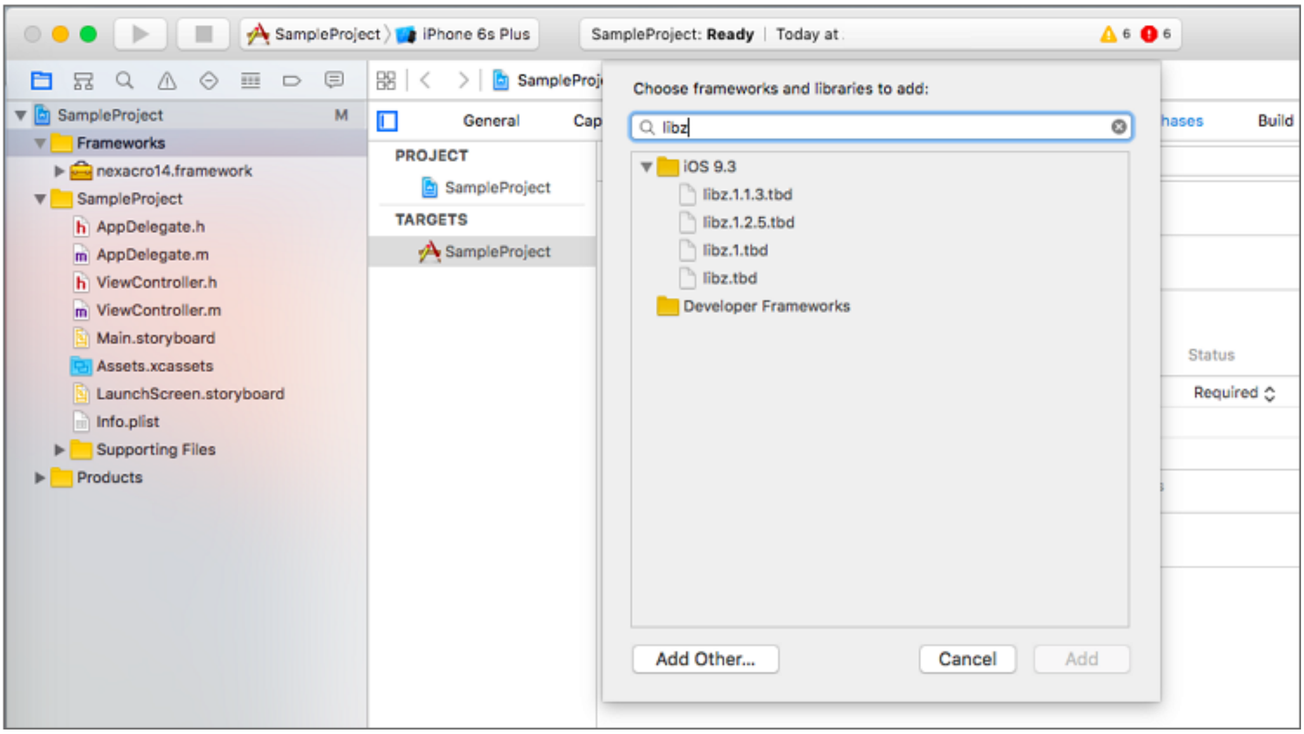
iOSが提供するAPIを使用するためのライブラリやフレームワークを設定します。Frameworksフォルダを選択> 右側のTARGETS項目を選択> 画面上部の「Build Phases」タブを選択> 「Link Binary With Libraries 」項目を選択すると、前項で追加したnexacro platformライブラリが設定されていることが確認できます。



下部の[+]ボタンをクリックし、以下の14個の項目を追加します。

libz.1.2.5.tbd
libsqlite3.0.tbd
MessageUI.fremawork
SystemConfiguration.framework
CoreMedia.framework
MobileCoreServices.framework
QuartzCore.fremawork
MediaPlayer.fremawork
CoreLocation.fremawork
CFNetwork.fremawork
AVFoundation.fremawork
AudioToolbox.framework
AddressBookUI.fremawork
AddressBook.fremawork

2015年6月以降のnexacro platformバージョンを使用する場合には、さらに以下の2つの項目を追加する必要があります。
Security.framework
libc++.tbd



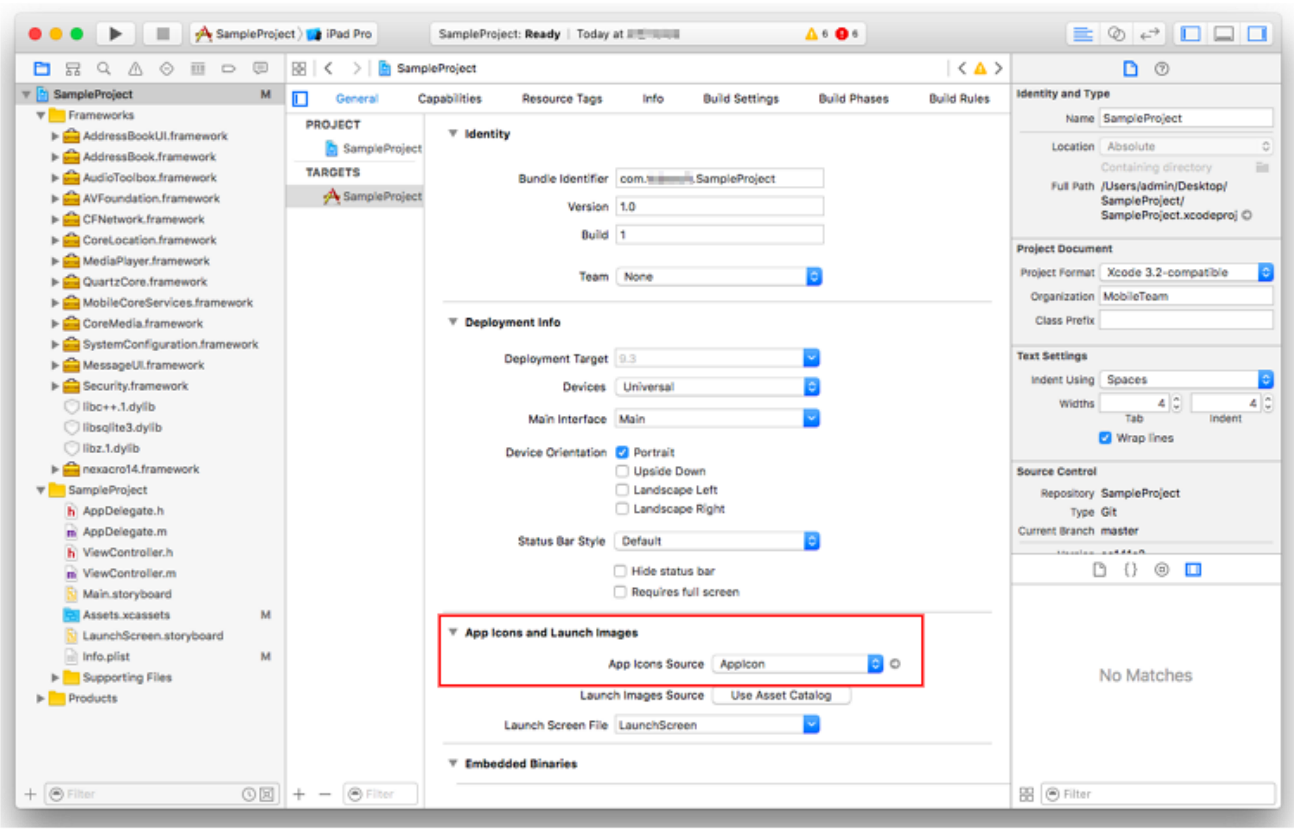
> 設定したライブラリ/フレームワークがプロジェクトの直下に追加された場合は、ドラッグ＆ドロップによりFrameworksフォルダの配下へ移動させます。

3.2.4　リソース設定

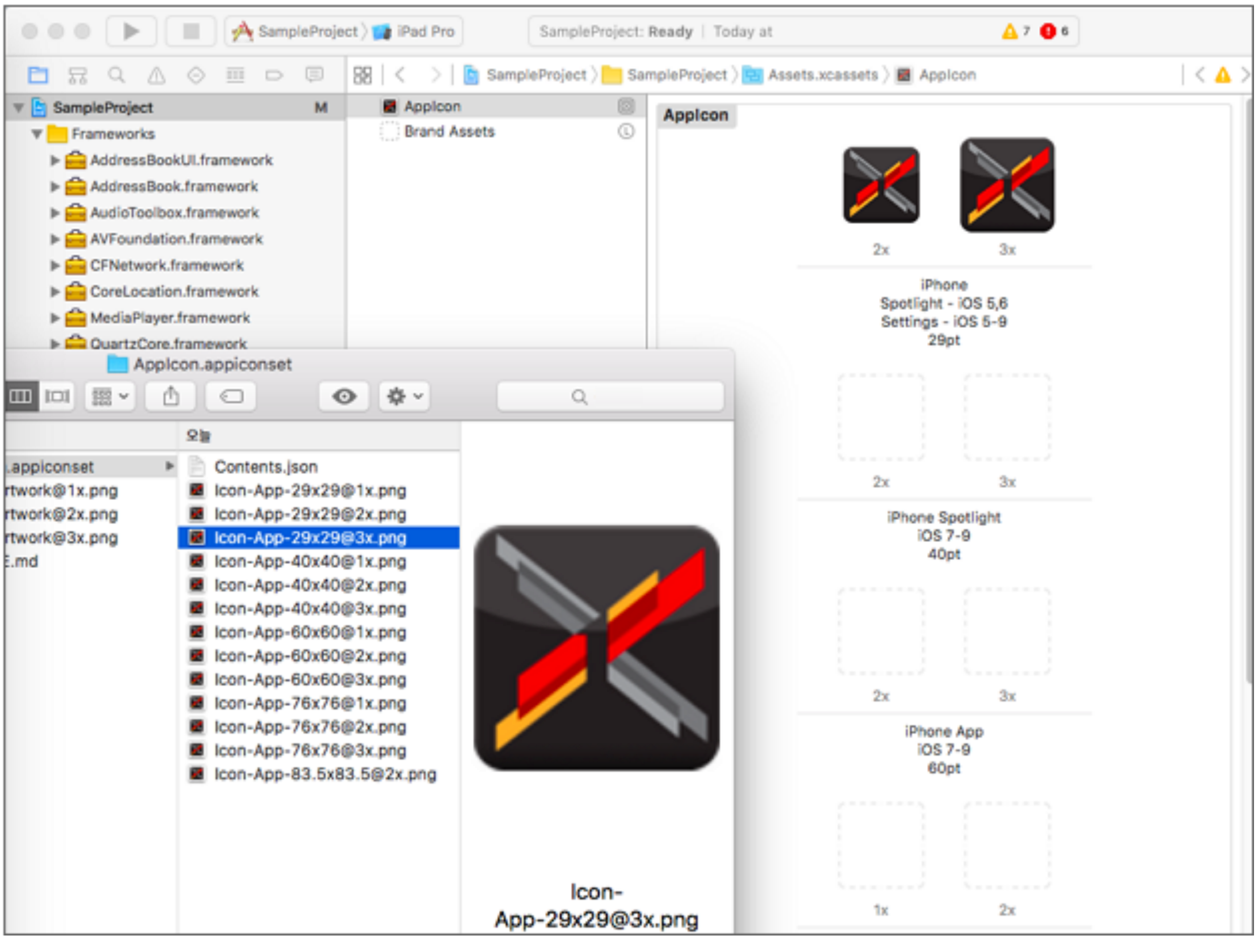
iOSアプリで使用するローディング画像、アイコン、メッセージ、レイアウトなどを設定するためのステップです。

アイコン画像の設定

アイコンおよびスプラッシュイメージは個別に指定します。あらかじめ作成したファイルをインポートすることができます。プロジェクトを選択> TARGETS項目を選択> 画面上部の「General」タブから設定します。



「App Icons and Launch Images」カテゴリーの項目「App Icons Source」で右側の矢印ボタンをクリックして表示される設定画面に、Finderからドラッグ&ドロップして、適切なサイズのファイルを指定します。

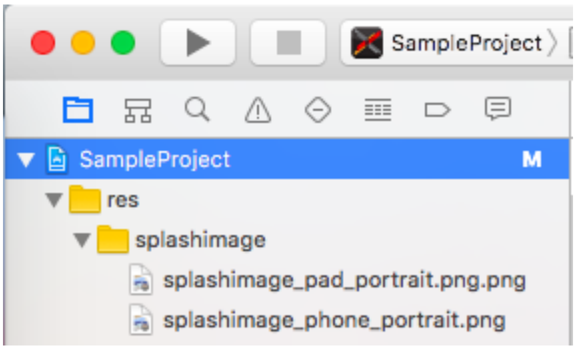


各項目に必要なアイコン画像のサイズは下記の通りです。

	1x	2x	3x
20pt (Notification icon)	20px x 20px	40px x 40px	60px x 60px
29pt (Settings icon)	29px x 29px	58px x 58px	87px x 87px
40pt (Spotlight icon)	40px x 40px	80px x 80px	120px x 120px
60pt (iphone icon)		120px x 120px	180px x 180px
76pt (iPad icon)	76px x 76px	152px x 152px	
83.5pt (iPad Pro icon)		167px x 167px	

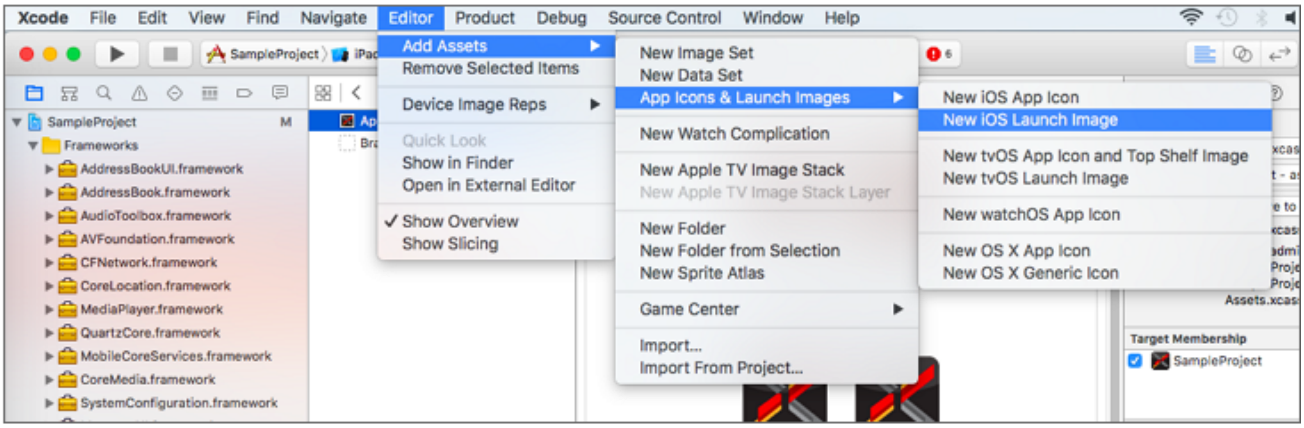
スプラッシュ画像の設定

スプラッシュ画像を使用するために、プロジェクトにフォルダを作成し、画像を追加します。単純な画像であれば、解像度ごとに画像を生成せずに一つのサイズ（750px x1334px）で対応することができます。

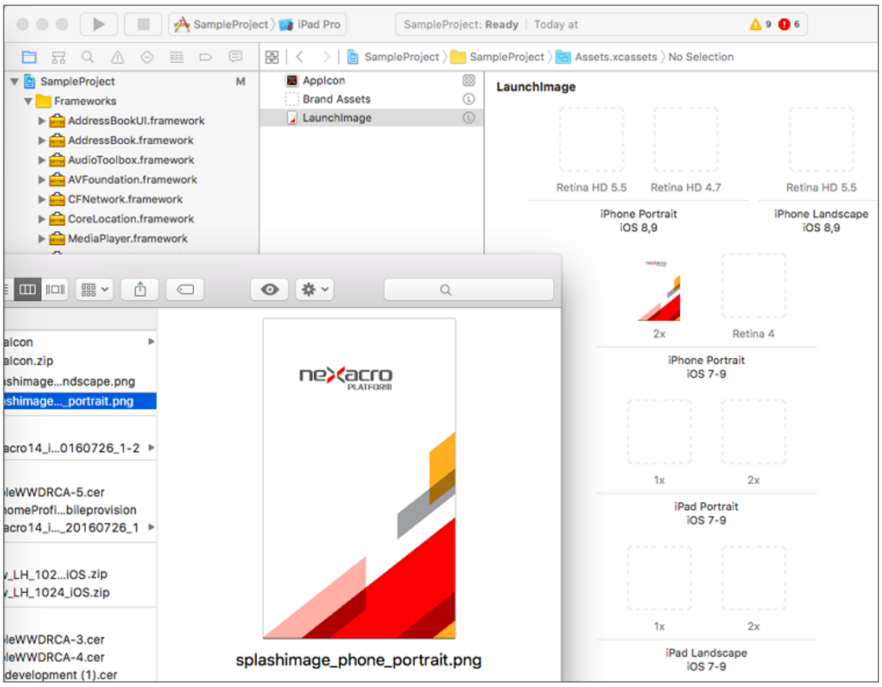


Xcodeで使用するスプラッシュ画像用のAsset Catalogを生成します。

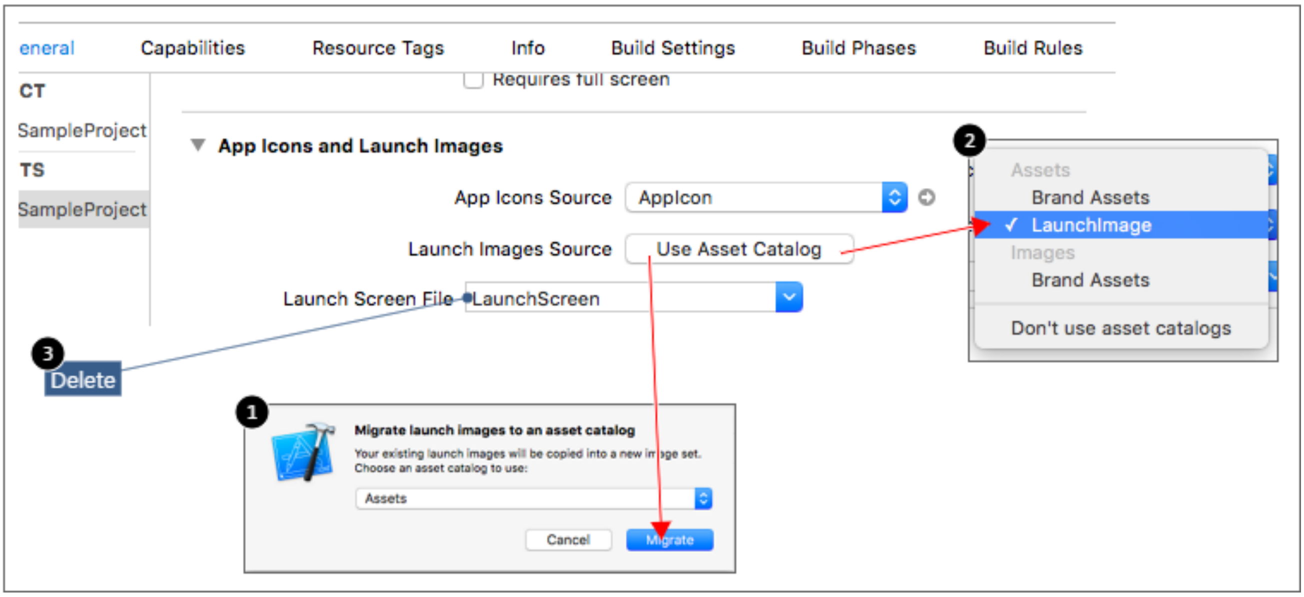
> Editor > Add Assets > App Icons & Launch Images > New iOS Launch Image

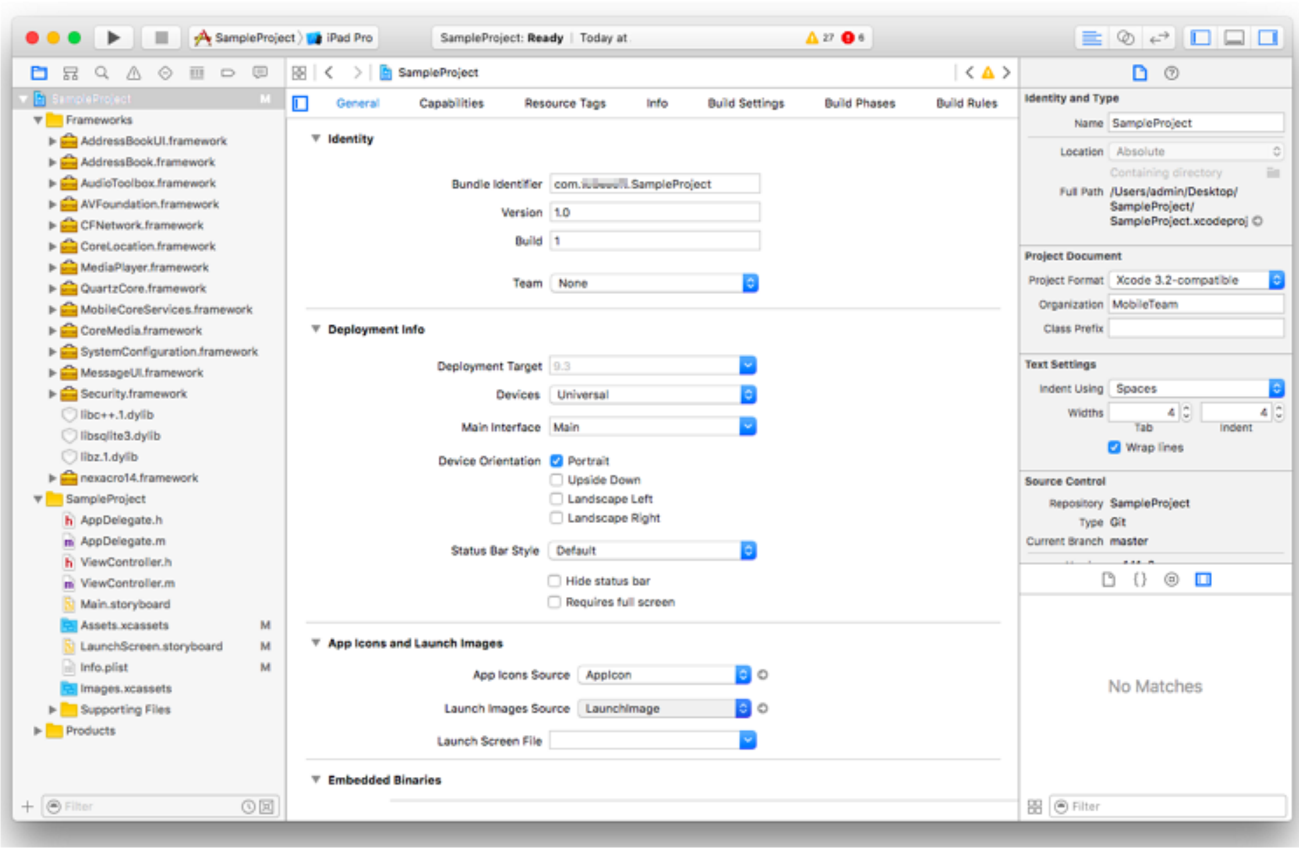


Asset Catalogの設定画面に、Finderからサイズに応じた各ファイルもしくは１つのファイル（750px x1334px）をドラッグ&ドロップして指定します。



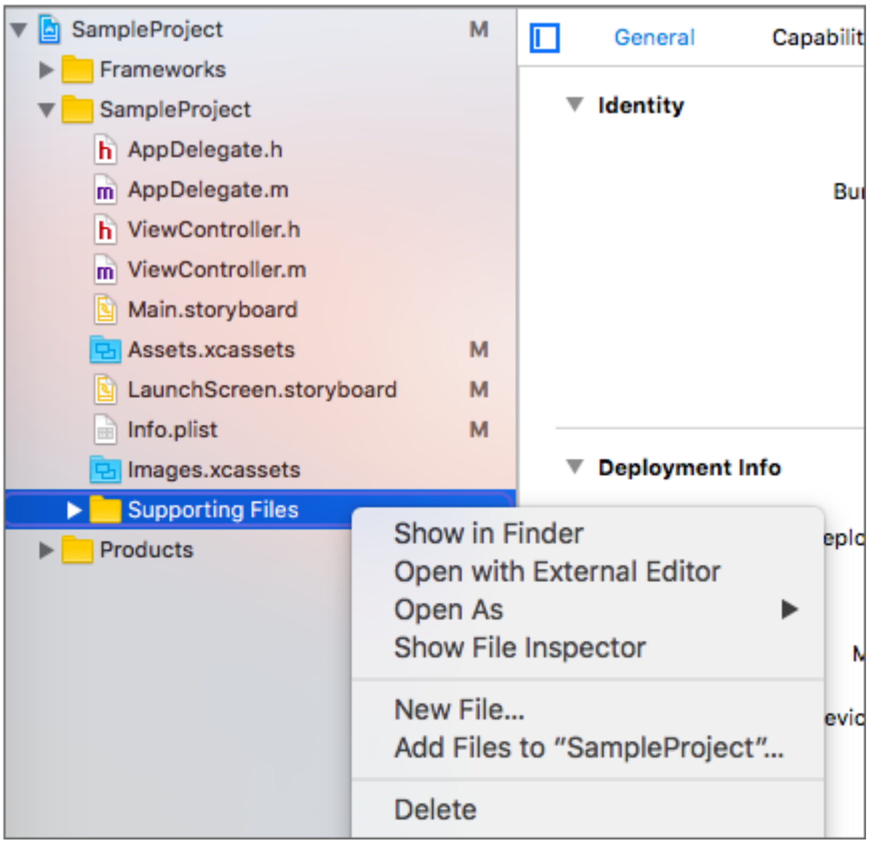
プロジェクトを選択> TARGETS項目を選択> 画面上部の「General」タブから設定します。「App Icons and Launch Images」カテゴリーの項目「Launch Images Source」で、「Use Asset Catalog」ボタンをクリックした後に表示される画面から[Migrate]ボタンをクリックします。「Launch Images Source」で新規作成したAsset Catalogを選択し、項目「Launch Screen File」の値は削除します。

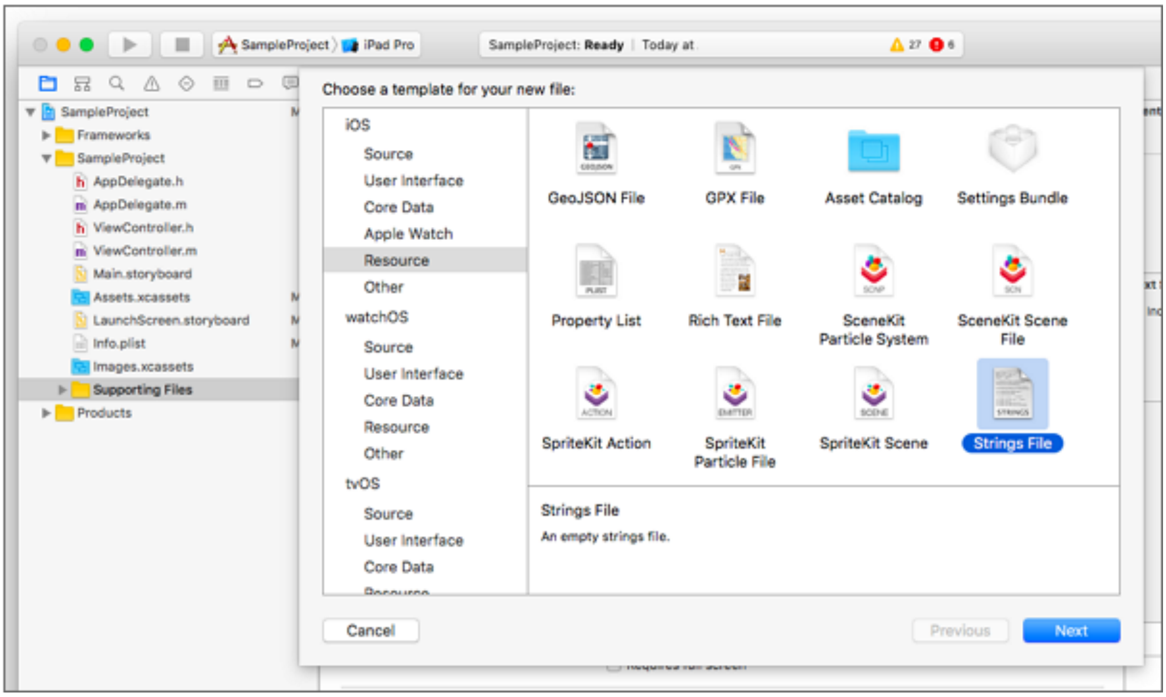




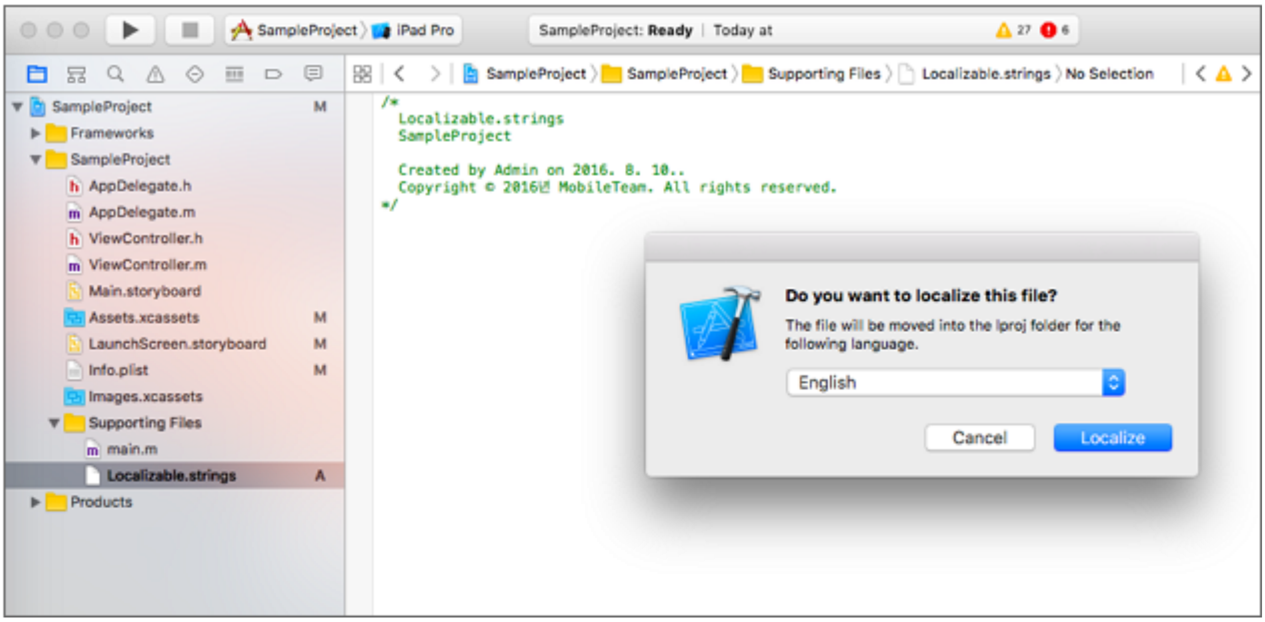
メッセージの設定

iOSアプリで使用されるメッセージを設定します。まず、Supporting Filesフォルダの配下に新しいファイルを追加します。フォルダを右クリック> コンテキストメニューから「New File」を選択し、テンプレートの中から「Strings File」を選択します。ファイル名は「Localizable」に指定します。

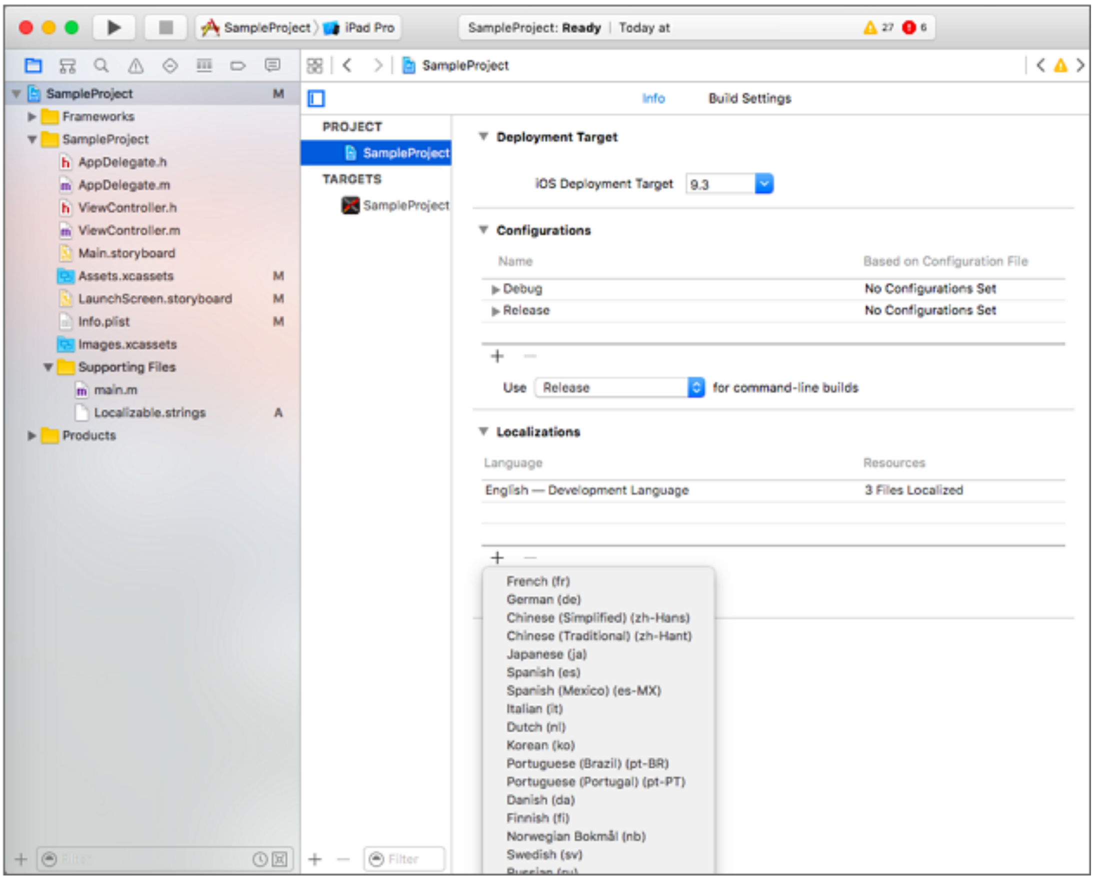




生成されたファイルを選択した状態で、右側のインスペクタペインから「Localization」カテゴリーの「Localize」ボタンをクリックし、希望する言語を選択します。



PROJECT項目を選択>「Info」タブを選択>「Localizations」カテゴリーに、上記で選択した言語が表示されます。必要に応じて、「+」ボタンにより複数の言語を追加することができます。追加された言語はプロジェクトフォルダからも確認することができます。



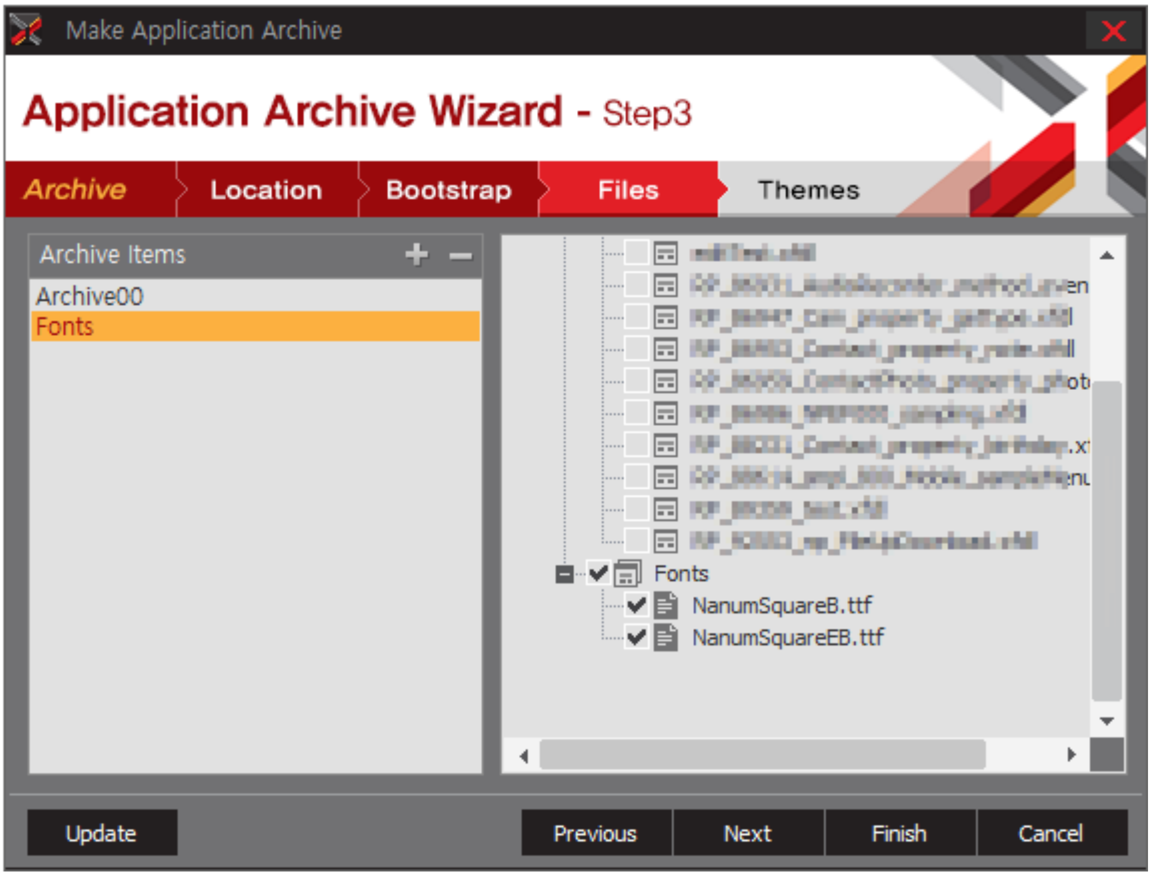
Localization.stringsファイルに対して、設定した言語に合わせたメッセージを編集します。以下は英語を設定した場合のサンプルです。

```
"needupdate" = "It is need to update.";
"loadingFail" = "First loading is fail.  \r\nPlease restart.";
"updateFail" = "Update is fail.  \r\nPlease restart.";
"notexist" = "Start file is NOT exist.  \r\nPlease restart.";
"BeingUpdated" = "Being updated.";
"wantreplace" = "There is a file of the save name. Do you want to replace? ";
"ok" = "OK";
"cancel" = "Cancel";
"move" = "Move";
"upper" = "Upper";
"filter" = "Filter";
"home" = "Home";
"nofilename" = "No File Name.";
"checkforupdates" = "Check for updates.";
"installforupdates" = "Install for updates.";
"downloadingforupdates" = "Downloading for updates.";
"force_close" = "Update is Completed.";
```

カスタムフォントの設定

> 14.0.1.3400以降のバージョンでは、WKWebView基盤で製品ライブラリが変更されました。
> 該当のバージョンでカスタムフォントの利用する場合には、下記の手順で適用することができます。

1. Xcodeで設定したフォントとplistに登録したフォントの設定を削除します。
2. nexacro studioでアーカイブファイルの生成時にフォントファイルを含むように構成します。



3. 生成されたRun.zipファイル内のRun.htmlファイルに、下記の通りに<style>タグを追加し再度圧縮ファイルを生成します。

srcの設定パスはnexacro studioでのアーカイブファイルの生成構造に合わせます。

```
<head>
...
    <style id="userfont">
    @font-face {
        font-family: 'NanumSquare';
        src: url(./Fonts/NanumSquareR.ttf) format('truetype')
    }
    @font-face {
        font-family: 'NanumSquare Bold';
        src: url(./Fonts/NanumSquareB.ttf) format('truetype')
    }
    </style>
...
</head>
```

3.2.5 コードファイルの設定

Xcodeプロジェクトを生成すると、デフォルトで生成されるAppDelegate.h、AppDelegate.m、main.mに対して、nexacro platform環境に合わせて編集します。

AppDelegate.h

生成されたAppDelegateがnexacro platformのAppDelegateを継承するように編集します。関連ファイルを次のように編集し、nexacro14.frameworkが動作するようにします。

```
#import <UIKit/UIKit.h>
#import <nexacro14/NexacroAppDelegate.h>
#import <nexacro14/NexacroMainViewController.h>
@interface AppViewController : NexacroMainViewController
-(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)
interfaceOrientation;
@end
@interface AppDelegate : NexacroAppDelegate <UIApplicationDelegate>
-(NexacroMainViewController*)initMainViewController:(NSInteger)nView;
@end
```

AppDelegate.m

以下のコードで、start_ios.jsonファイルを指定した部分に対して、nexacro studioで作成したjsonファイルが配置されているサーバーのURLに書き替えます。

```
#import "AppDelegate.h"
@implementation AppViewController
-(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)
interfaceOrientation
{
    //return NO;
    return [super shouldAutorotateToInterfaceOrientation:interfaceOrientation];
}
@end
@implementation AppDelegate
-(NexacroMainViewController*)initMainViewController:(NSInteger)nView
```

```
{
    [[NexacroResourceManager sharedResourceManager]setBootstrapURL:@"http
://172.10.10.10:8080/nexacro/start_ios.json"];
    return [[AppViewController alloc] initWithLoader:nView];
}
@end
```

main.m

Supporting Filesフォルダの配下にあるmain.mファイルを以下のように編集します。

```
#import <UIKit/UIKit.h>
#import "AppDelegate.h"
int main(int argc, char *argv[])
{
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([
AppDelegate class]));
    }
}
```

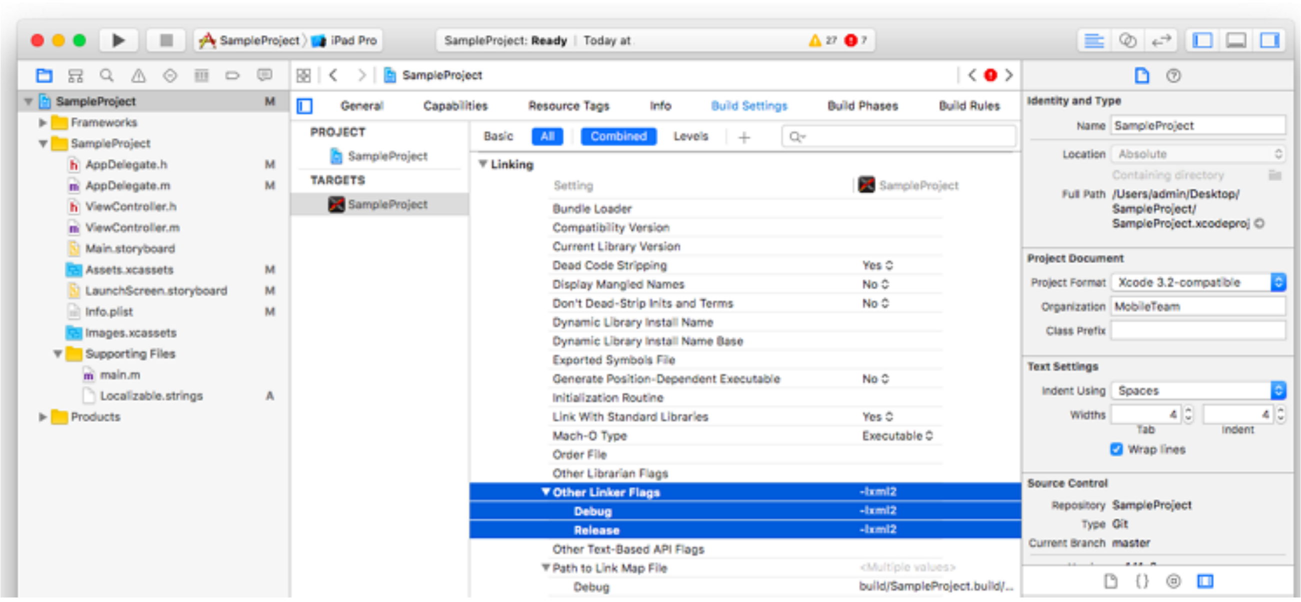
3.2.6　ビルド環境の設定

Build Settings

プロジェクトを選択> TARGETSを選択> 画面上部の「Build Settings」タブから一部の設定を変更する必要があります。

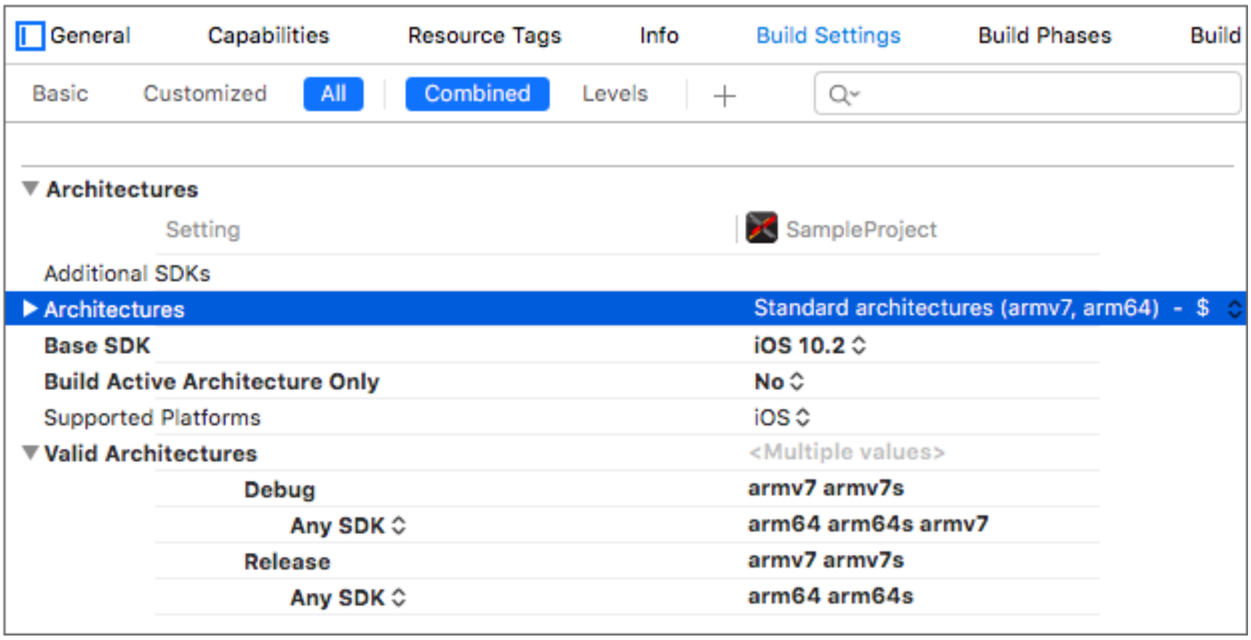
Build Settings > Linking > Other Linker Flags	
Debug	-lxml2
Release	-lxml2
Build Settings > Search Paths > Header Search Paths	
Debug	/usr/include/libxml2
Release	/usr/include/libxml2
Build Settings > Architectures > Build Active Architecture Only	

Build Settings > Linking > Other Linker Flags	
Debug	No
Release	No
Build Settings > Architectures > Vaild Architectures	
Debug	armv7 armv7s
Release	armv7 armv7s
Build Settings > Apple LLVM *.* - Language - objective C	
Objective-C Automatic Reference Counting	No
Build Settings > Build Options	
Enable Bitcode	No



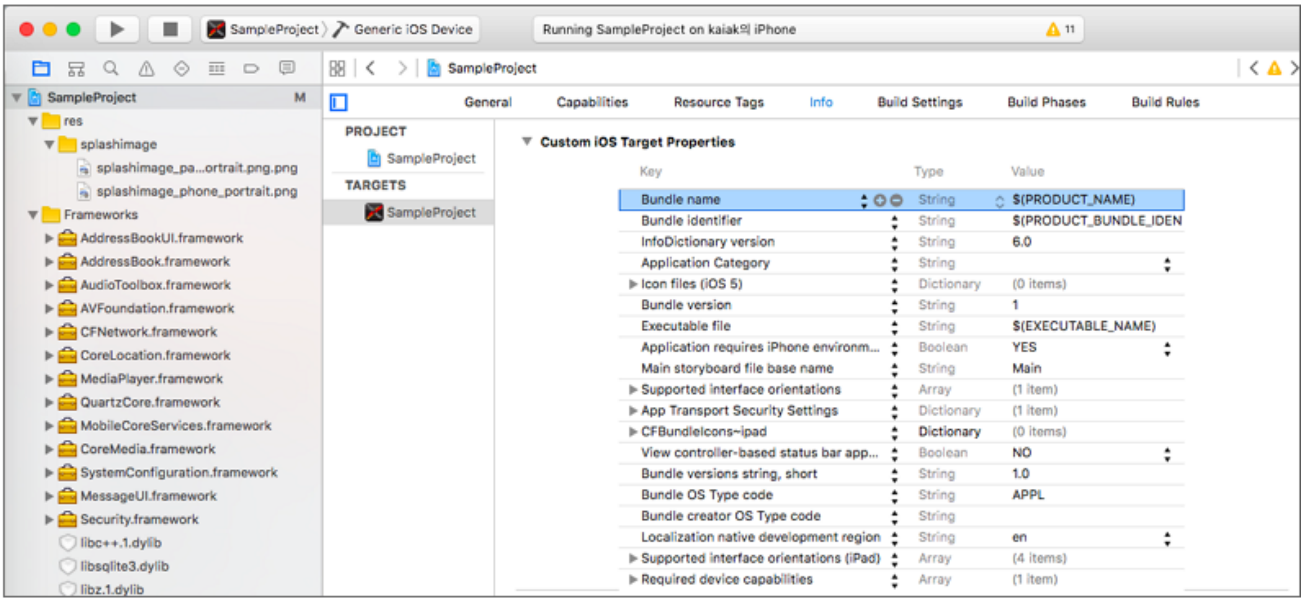
iOSのバージョンアップにより、ユーザーの同意後にアクセス可能な機能に関する権限要求のための設定や機能追加による設定が必要な場合があります。iOSのバージョンに応じて、Architectures項目を追加で設定します。

> ARMv7 : iPhone 3GS, iPad(2010), iPhone 4, iPod touch, iPad2, iPhone 4S, (new)iPad, iPad mini
> ARMv7s : iPhone 5, iPad(2012), iPhone 5c
> ARM64 : iPhone 5S, iPad Air, iPad mini 2, iPhone 6, iPhone 6 plus, iPad Air 2, iPad mini3



Info

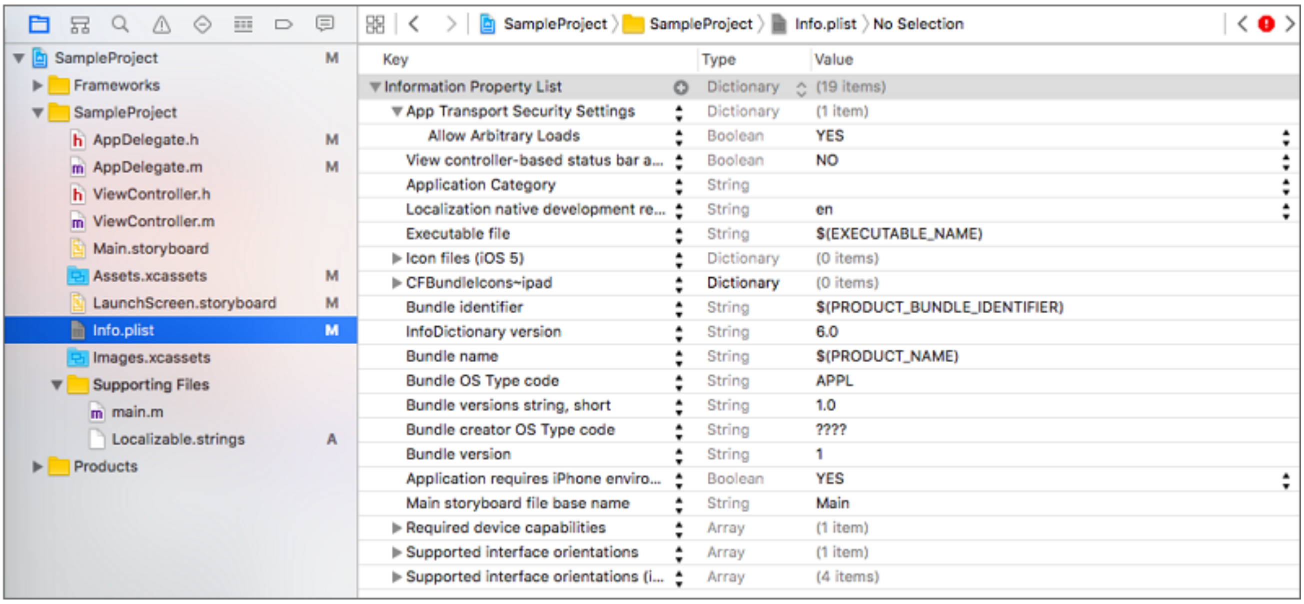
プロジェクトを選択> TARGETSを選択> 画面上部のInfoタブから、設定内容を確認します。Bundle Nameに指定された値はアプリの名前になり、Bundle identifierに指定された値はアプリを識別する値になります。



Info.plist

プロジェクトからInfo.plistファイルを選択し、一部の設定を変更します。「Information Property List」>「App Transport Security Settings」項目を追加>「Allow Arbitrary Loads」項目を追加し、値を「YES」に設定します。そして、「Information Property List」>「View controller-based status bar appeara

nce」項目を追加し、値を「NO」に設定します。



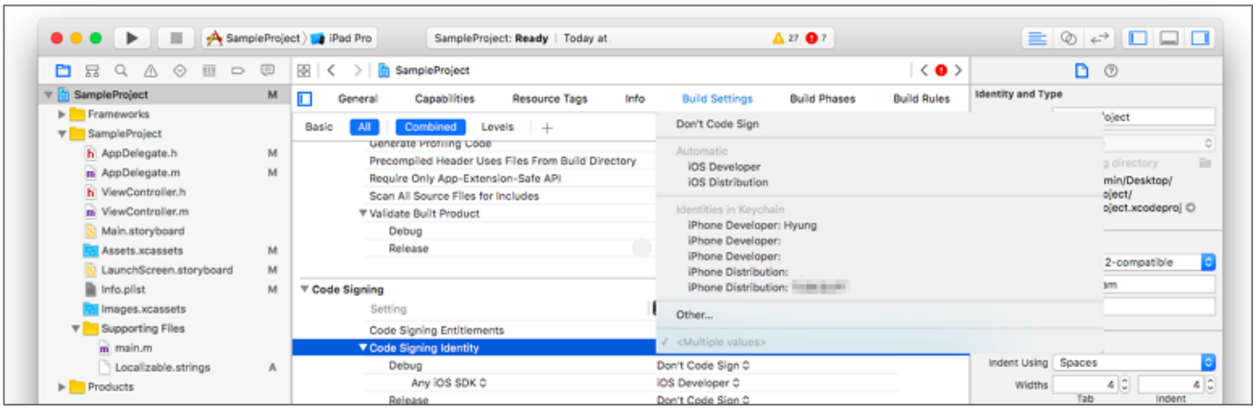
位置情報の使用、カメラおよびフォトアルバムへのアクセスに関する設定をinfo.plistに追加します。

Privacy - Camera Usage Description	String	
Privacy - Location Always Usage Description	String	used for geolocation
Privacy - Location Usage Description	String	used for geolocation
Privacy - Location When In Use Usage Description	String	used for geolocation
Privacy - Photo Library Usage Description	String	

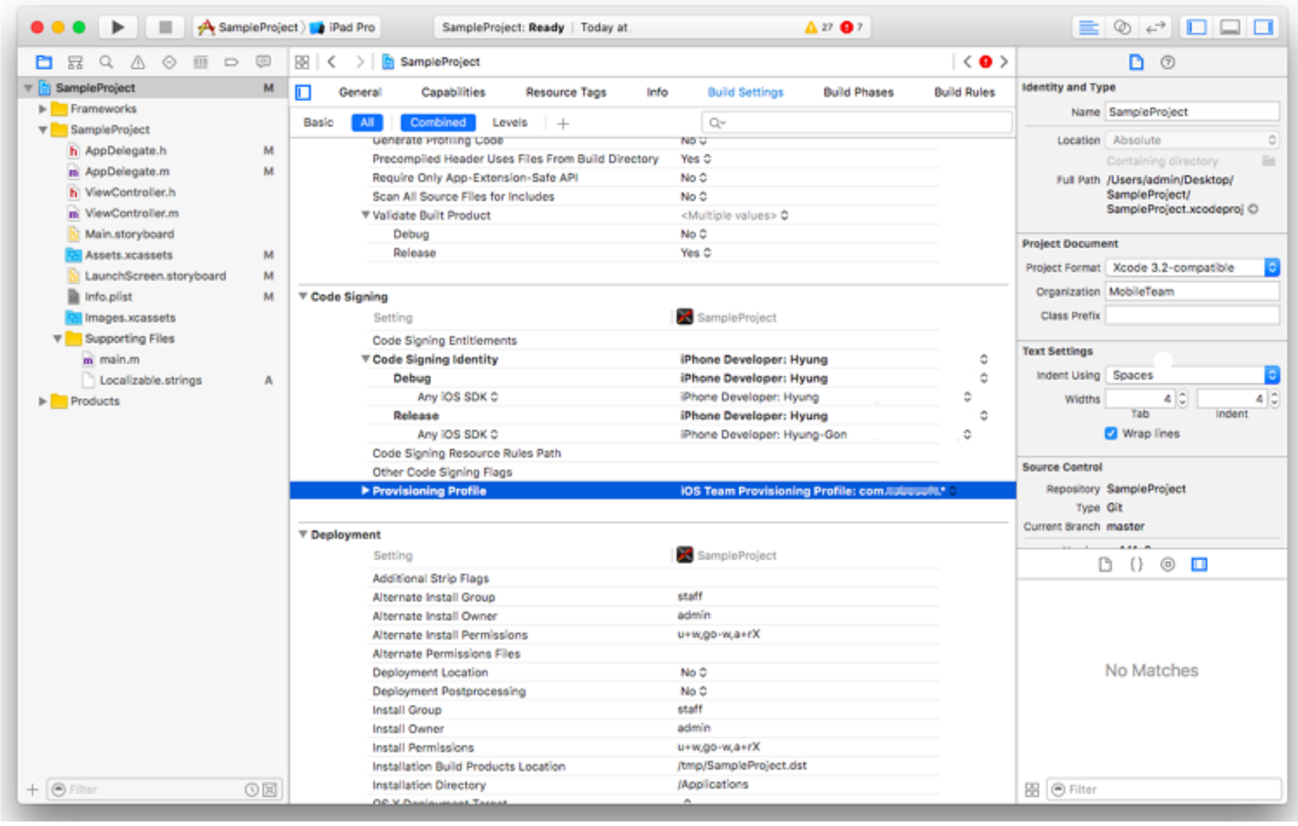
Code Singing

Xcodeのシミュレータでテストする場合は別途認証手続きが不要ですが、実機にアプリを転送してテストする場合は追加設定が必要です。

プロジェクトを選択> TARGETSを選択> 「Build Settings」タブの項目「Code Signing」の「Code Signing Identity」に対して、以下のように適切なプロファイルを設定します。



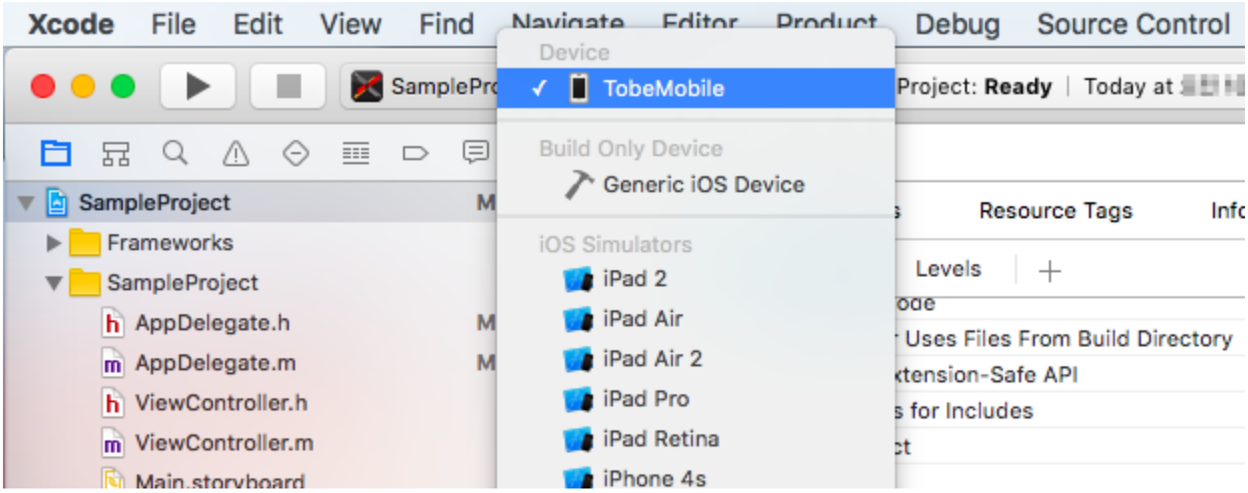
そして、「Provisioning Profile」に対して、プロビジョニングプロファイルを設定します。



使用可能な証明書はキーチェーンで確認することができ、プロビジョニングプロファイルはXcode（Preferences>Accounts）で確認することができます。

3.3　アプリのテスト

1. iOS端末をUSBでPCと接続して実機テストを行うか、もしくはXcodeのシミュレータを立ち上げてテストを行うことができます。実機テストを行う場合には、端末にプロファイルをインストールする必要があります。



2. Xcode上で、テストを行う対象を選択し、Run機能により実行します。画面上部のアイコンをクリック、もしくは下記のメニューから実行することができます。

```
Product > Run
```

第4章

アプリ開発と実行(Android)

4.1　アプリ開発環境設定

Androidアプリは、Eclipse、Android SDKを利用して開発し、基本的なステップは、一般的なAndroidアプリを作成する方法と同じです。

EclipseとAndroidのSDKインストールファイルの関連資料は、GoogleのAndroid Webページを参照してください。

> http://developer.android.com/sdk/installing.html

EclipseのとAndroidのSDKは最新バージョンの更新サイクルが早いので、上記のWebページのリンクを利用して最新の開発環境にインストールする必要があります。インストールの詳細は、Eclipse、ADT、Android SDKのバージョンが更新されるたびに少しずつ変更されることがあります。

アンドロイド基本開発環境は、基本的に次のような手順で進行します。

1. JDK(Java SE Development Kit) インストール
2. Eclipse インストール
3. Android SDK インストール
4. ADT プラグインインストール

> 2014年12月、Googleから新しい開発ツールであるAndroid Studio（アンドロイドスタジオ）が公開されました。
> https://en.wikipedia.org/wiki/Android_Studio
> Android Studioを使用する方法と、既存の EclipseのプロジェクトをAndroid Studioにインポートする方法を、この章に追加しました。

2015年6月から、Googleは Eclipse ADTプラグインのサポートを中止しました。
http://developer.android.com/tools/help/adt.html
開発環境に応じて、Android Studioのみをサポートすることもあります。

既にアプリの開発環境が設定されている場合はアプリ開発環境の設定は省略可能です。

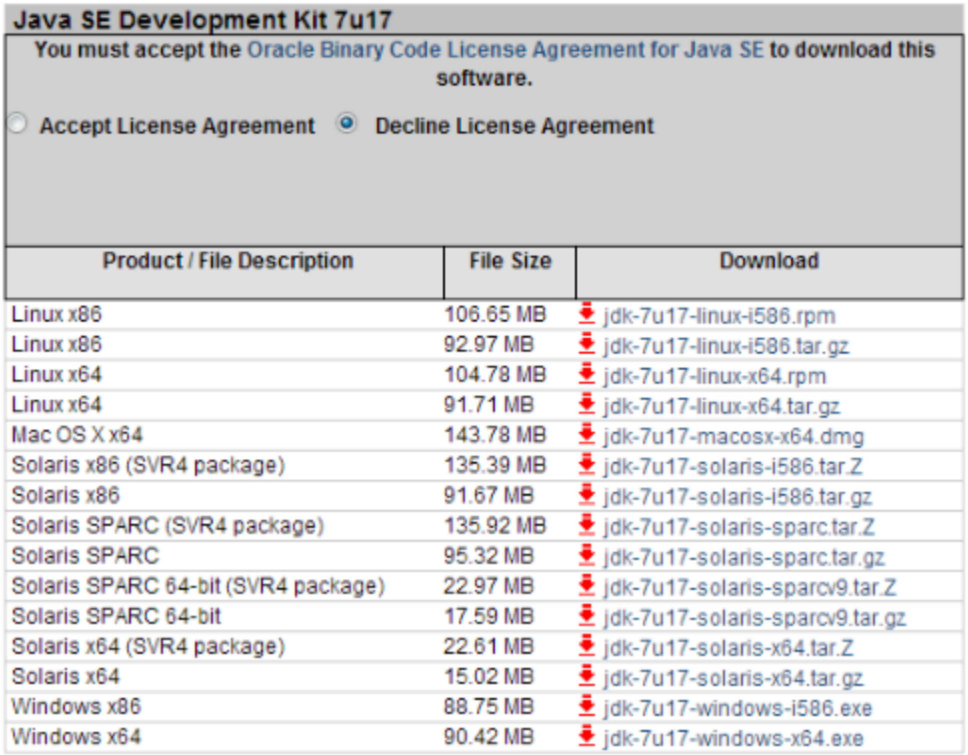
4.1.1　JDK(Java SE Development Kit) インストール

Android Studioと Eclipseは Javaベースで作られているので、実行するには JDK(Java SE Development Kit)環境が設定されている必要があります。JDKは、無料で提供されており、簡単にインストールできます。

ダウンロード

下記のサイトで、システム環境に合ったJDKをダウンロードすることができます。

http://www.oracle.com/technetwork/java/javase/downloads/index.html



Java SE Development Kit 7u17

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement　Decline License Agreement

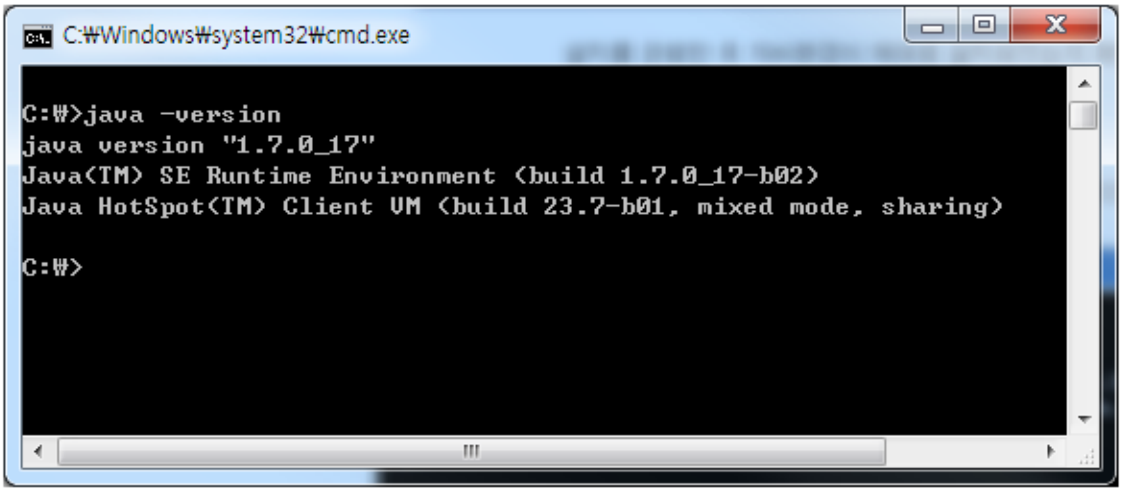
Product / File Description	File Size	Download
Linux x86	106.65 MB	jdk-7u17-linux-i586.rpm
Linux x86	92.97 MB	jdk-7u17-linux-i586.tar.gz
Linux x64	104.78 MB	jdk-7u17-linux-x64.rpm
Linux x64	91.71 MB	jdk-7u17-linux-x64.tar.gz
Mac OS X x64	143.78 MB	jdk-7u17-macosx-x64.dmg
Solaris x86 (SVR4 package)	135.39 MB	jdk-7u17-solaris-i586.tar.Z
Solaris x86	91.67 MB	jdk-7u17-solaris-i586.tar.gz
Solaris SPARC (SVR4 package)	135.92 MB	jdk-7u17-solaris-sparc.tar.Z
Solaris SPARC	95.32 MB	jdk-7u17-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	22.97 MB	jdk-7u17-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	17.59 MB	jdk-7u17-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	22.61 MB	jdk-7u17-solaris-x64.tar.Z
Solaris x64	15.02 MB	jdk-7u17-solaris-x64.tar.gz
Windows x86	88.75 MB	jdk-7u17-windows-i586.exe
Windows x64	90.42 MB	jdk-7u17-windows-x64.exe

この章では、jdk-7u17-windows-i586.exe (version 1.7.0_17)ファイルをダウンロードして使用しました。サイトの更新により、一部内容が変更されることがあります。

インストール確認

ダウンロードしたインストールファイルを実行すると、自動的に必要な環境設定とインストールを行います。インストールが完了したら、コマンドウィンドウで、正常にインストールされていることを確認します。

```
java -version
```



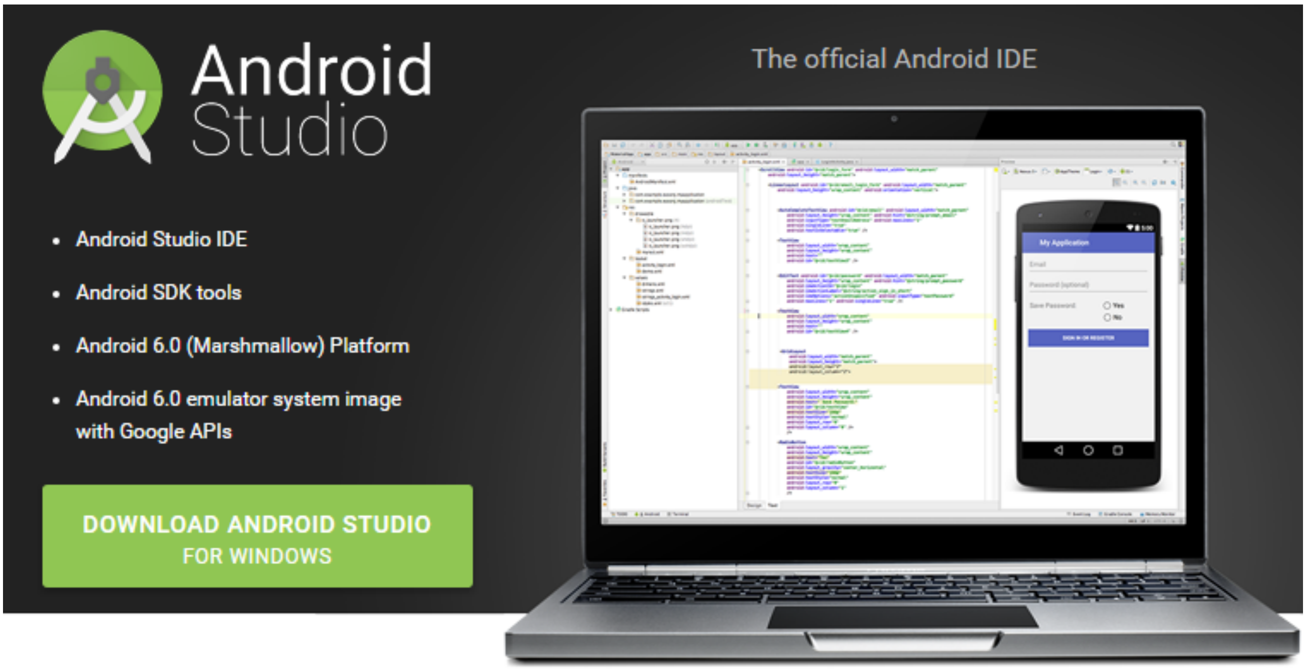
JDK1.5以降のバージョンをインストールする場合には、実行に必要な環境変数(JAVA_HOME)を別々に設定しなくても、システムパスに自動的に実行ファイルをコピーしてアクセスすることができます。

4.1.2 開発ツールインストール

Android Studio

Android Studioのインストールファイルは、基本的に SDKツールが含まれています。下記のサイトから環境に合ったインストールファイルをダウンロードすることができます。

> http://developer.android.com/sdk/index.html



> Android Studioをデフォルトのオプションでインストールした場合には、すぐに開発段階に進みます。

> この章では、Android Studio 1.5.1バージョンのインストールを基準に作成しました。

Eclipse

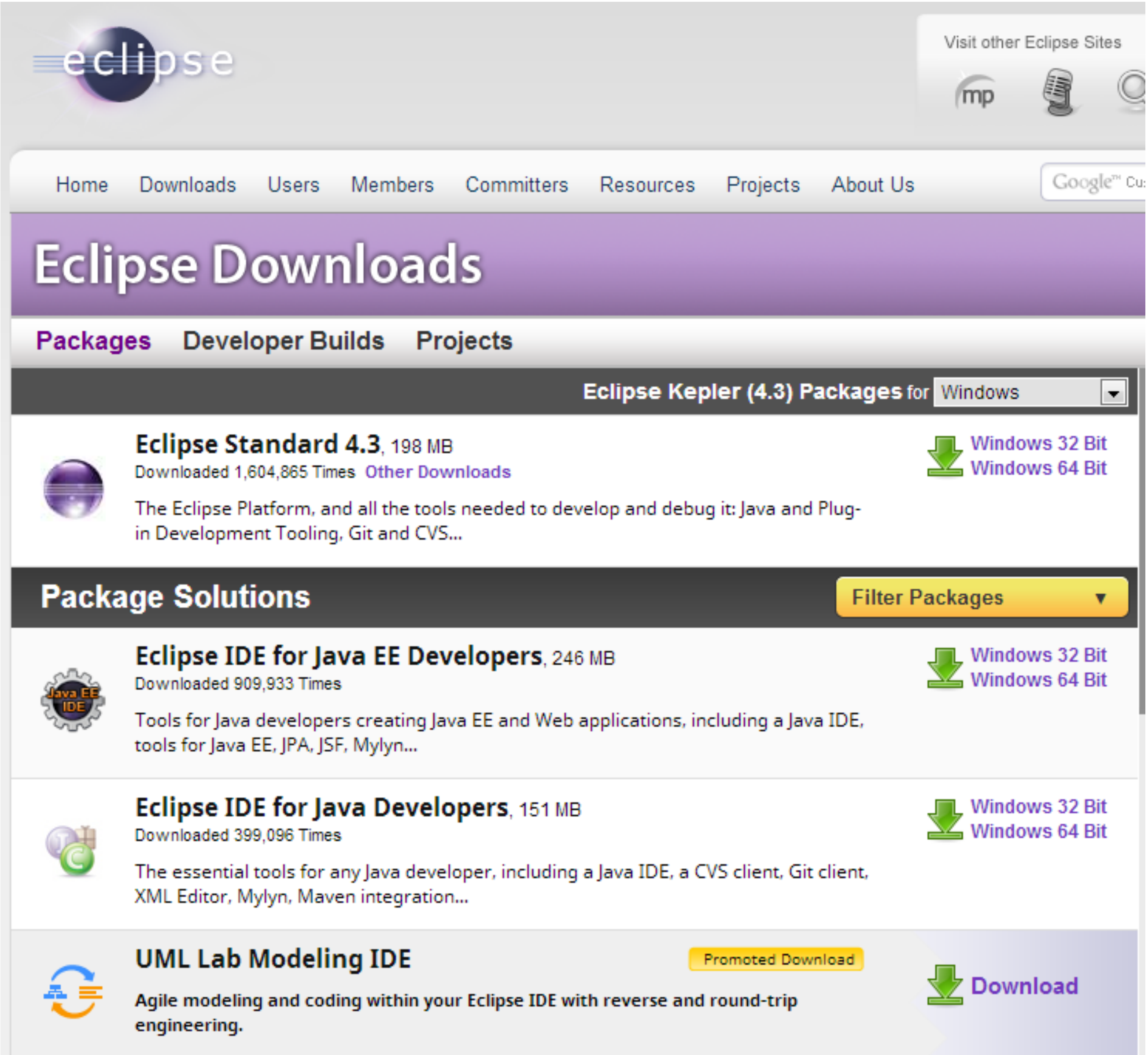
> nexacro platform 14.0.1.3000以降のバージョンからFCMサービス機能が追加され、Eclipseをサポートしなくなりました。Android Studioでのみアプリプロジェクトをビルドすることができます。

> EclipseのADTプラグインは2015年にサポートが終了し、その後追加されたサービス機能を使用する際に制約がありました。
> https://developer.android.com/studio/tools/sdk/eclipse-adt

Eclipseは、個別のインストール過程なしにファイルをダウンロードして解凍すればすぐに使用できます。

下記のサイトでは、環境に合ったファイルをダウンロードすることができます。 Eclipse Standard版のダウンロードまたは開発環境に合ったファイルを選択してダウンロードします。

> http://www.eclipse.org/downloads/

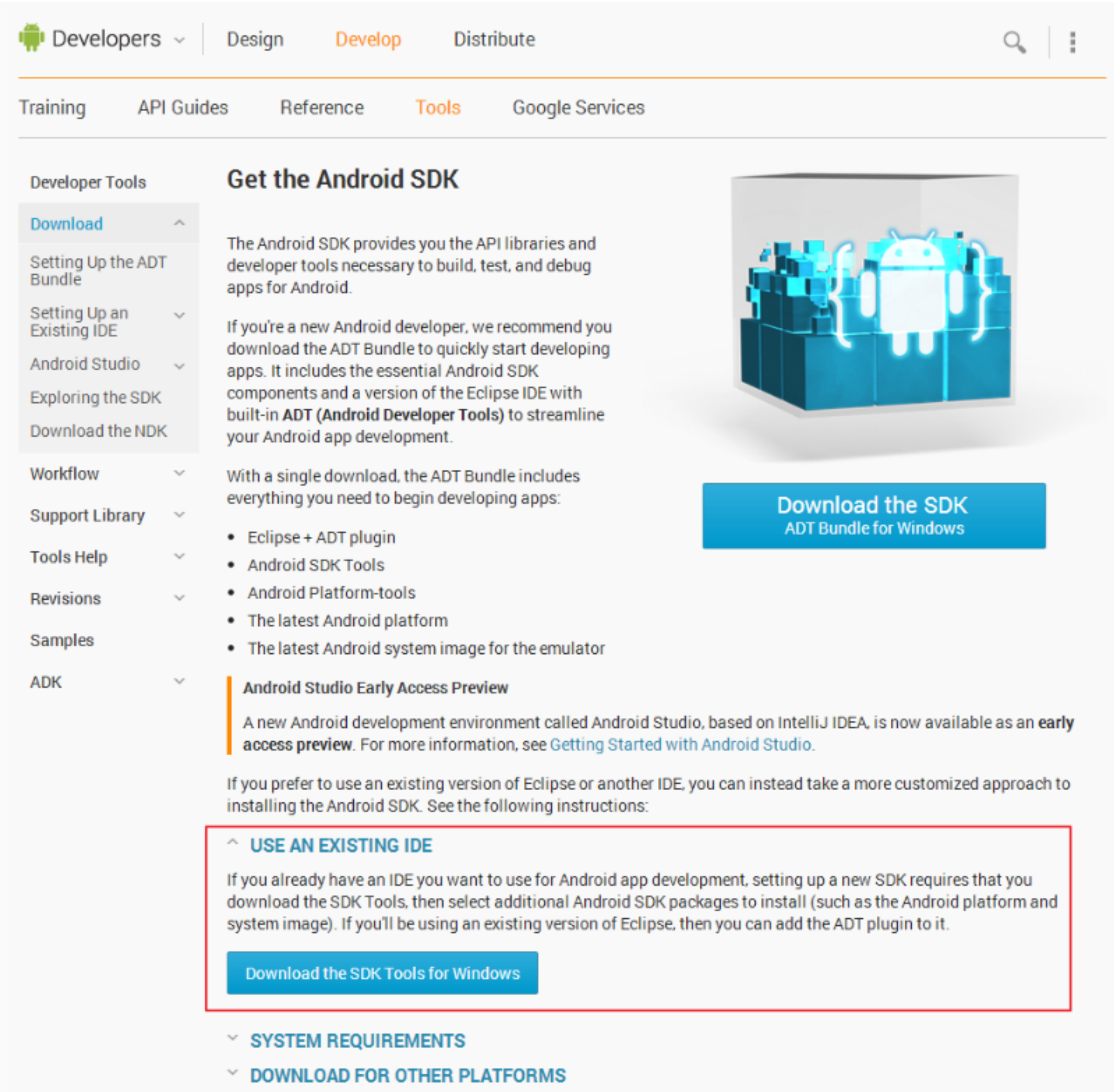


Android SDK インストール

Androidのプロジェクトを進行するためには関連したAPIを提供するAndroid SDKをインストールする必要があります。該当ファイルは、Androidの開発者サイトで提供しています。

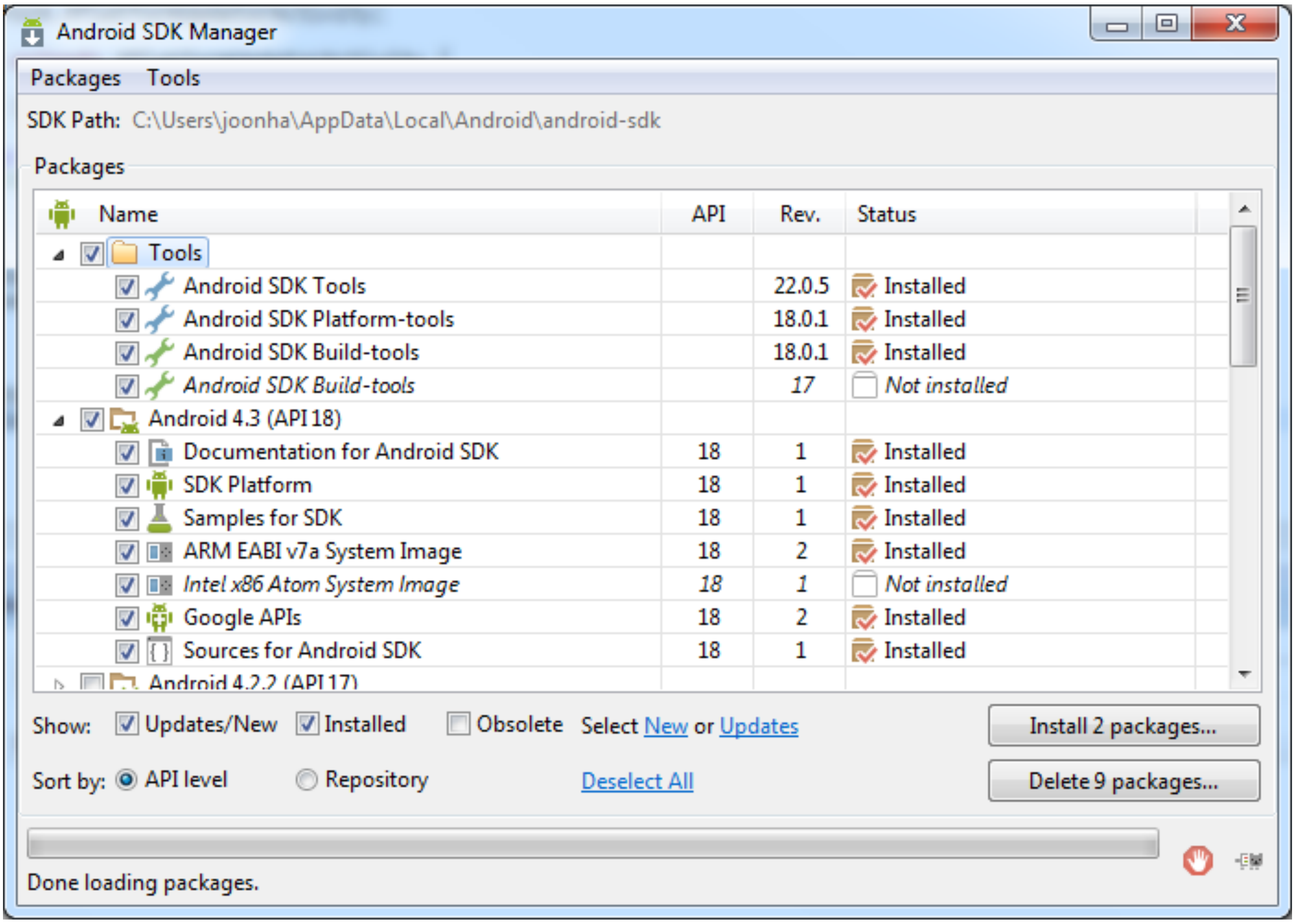
ADTプラグインがインストールされたEclipseを含むADT Bundleパッケージをダウンロードしたり、SDKインストールファイルを個別にダウンロードできます。SDKインストールファイルのみダウンロードするには、以下のサイトの下のメニューから「USE AN EXISTING IDE」リンクを選択し、「SDK Tools for Windows」インストールファイルをダウンロードします。

http://developer.android.com/sdk/index.html



インストールファイルを実行してインストールが完了すると、SDK Managerが実行されます。SDK Managerは、開発に必要なAndroidのAPIと関連ツールをダウンロードできる機能を提供します。項目の中で「Tools」、「Extras」の項目を選択し、必要なAndroid APIを選択してインストールします。

SDK Managerは、必要によって再実行して、追加的にAPIをインストールすることができます。

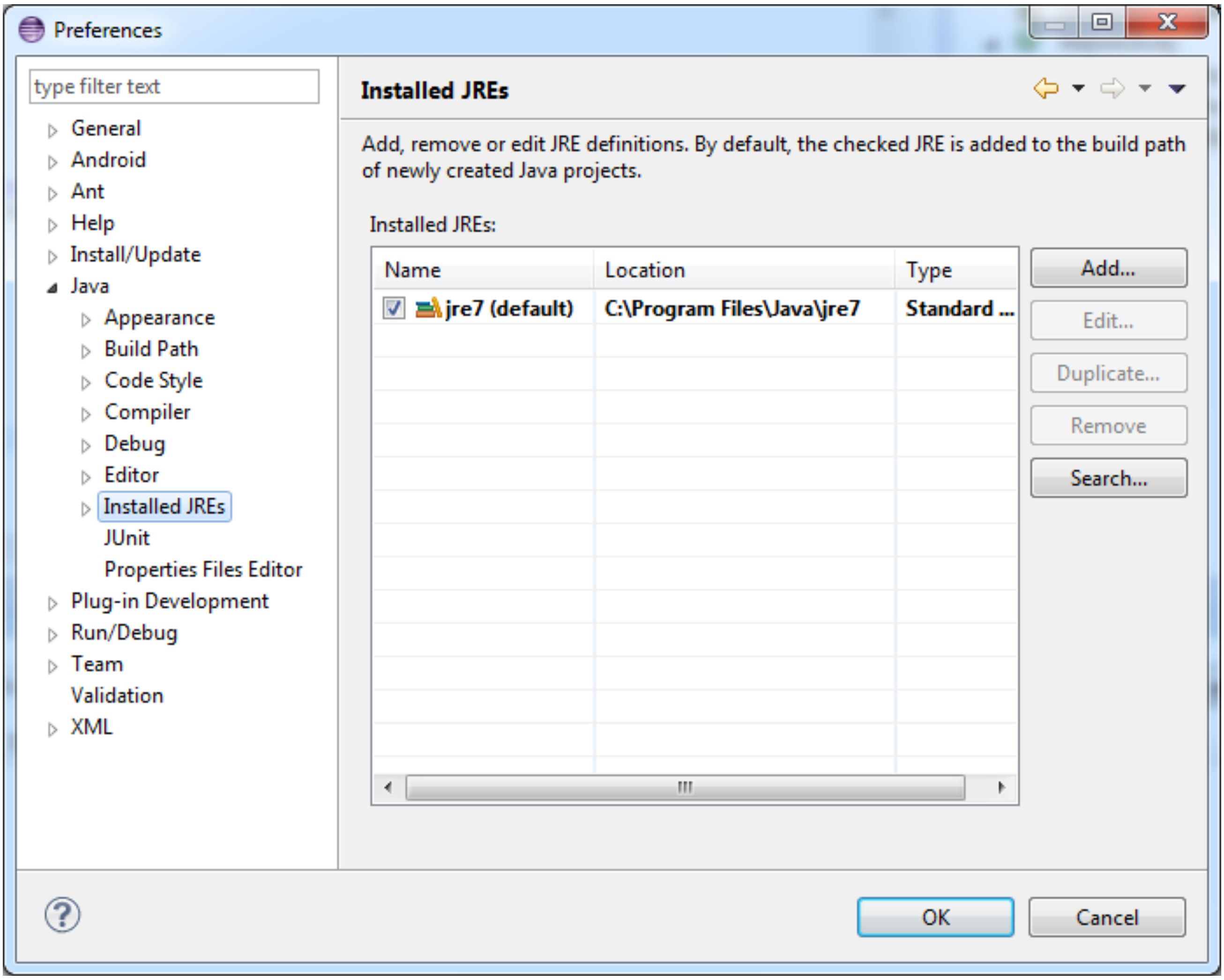


Android APIのインストールは、ネットワーク環境によって時間がかかることがあります。

ADTプラグインのインストール

Eclipseを実行すると、インストールされたJDKのバージョンに該当するJRE(Java Runtime Environment)が設定されていることを確認します。JREの設定は、以下のメニューで確認できます。

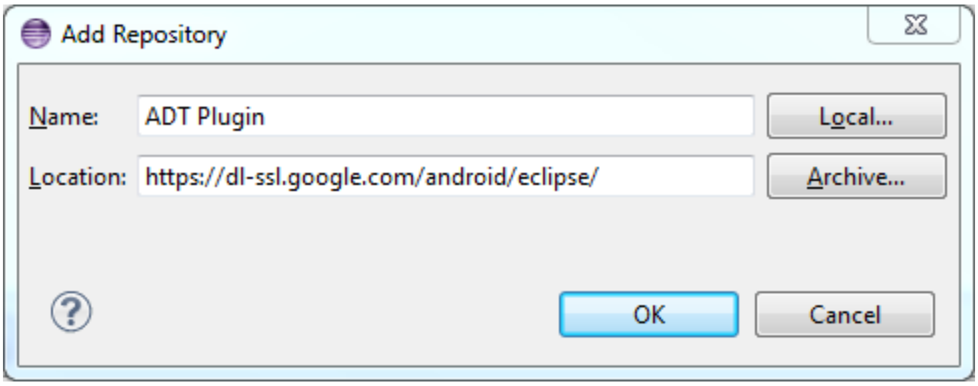
```
Window > Preference > Java > Installed JREs
```



Androidのプロジェクトを実行するためにADTプラグインをインストールします。下のメニューに移動してDeveloper Toolsを追加します。

```
Help > Install New Software
```

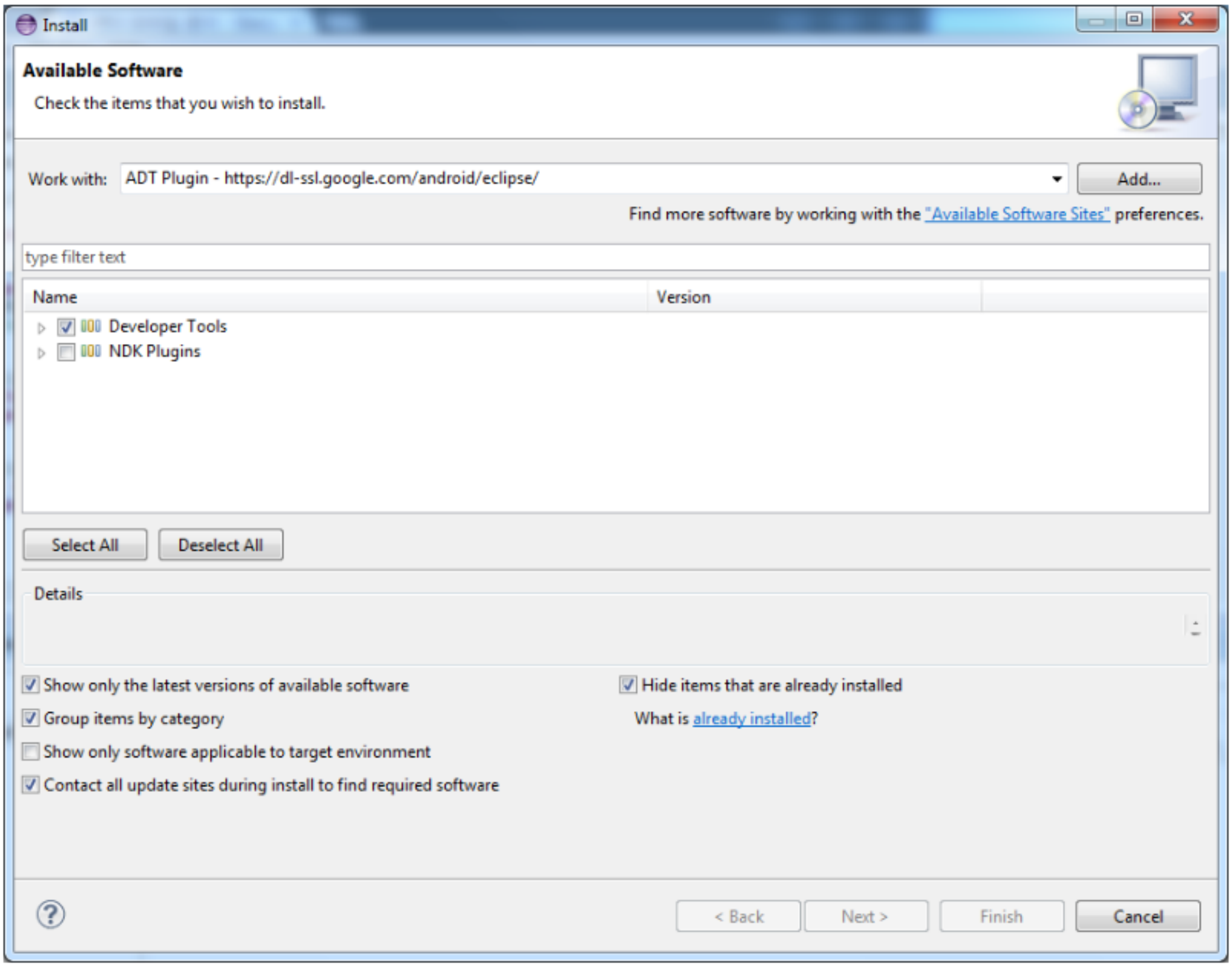
> 使用しているEclipseのバージョンによって、追加機能をインストールするためのメニューが異なります。Eclipse 3.4以前のバージョンである場合には、[Help > Software Updates] メニューから確認できます。
> この章では、Eclipse4.3(Kepler)バージョンを使用しています。



Work withの項目の横にある「Add」ボタンをクリックして、次のようにRepositoryの情報を入力します。

```
ADT Plugin
https://dl-ssl.google.com/android/eclipse/
```

選択できる項目から「Developer Tools」を選択してインストールを進行します。インストールが完了すると、Eclipseが再起動されます。



4.1.3　Eclipseプロジェクトのマイグレーション

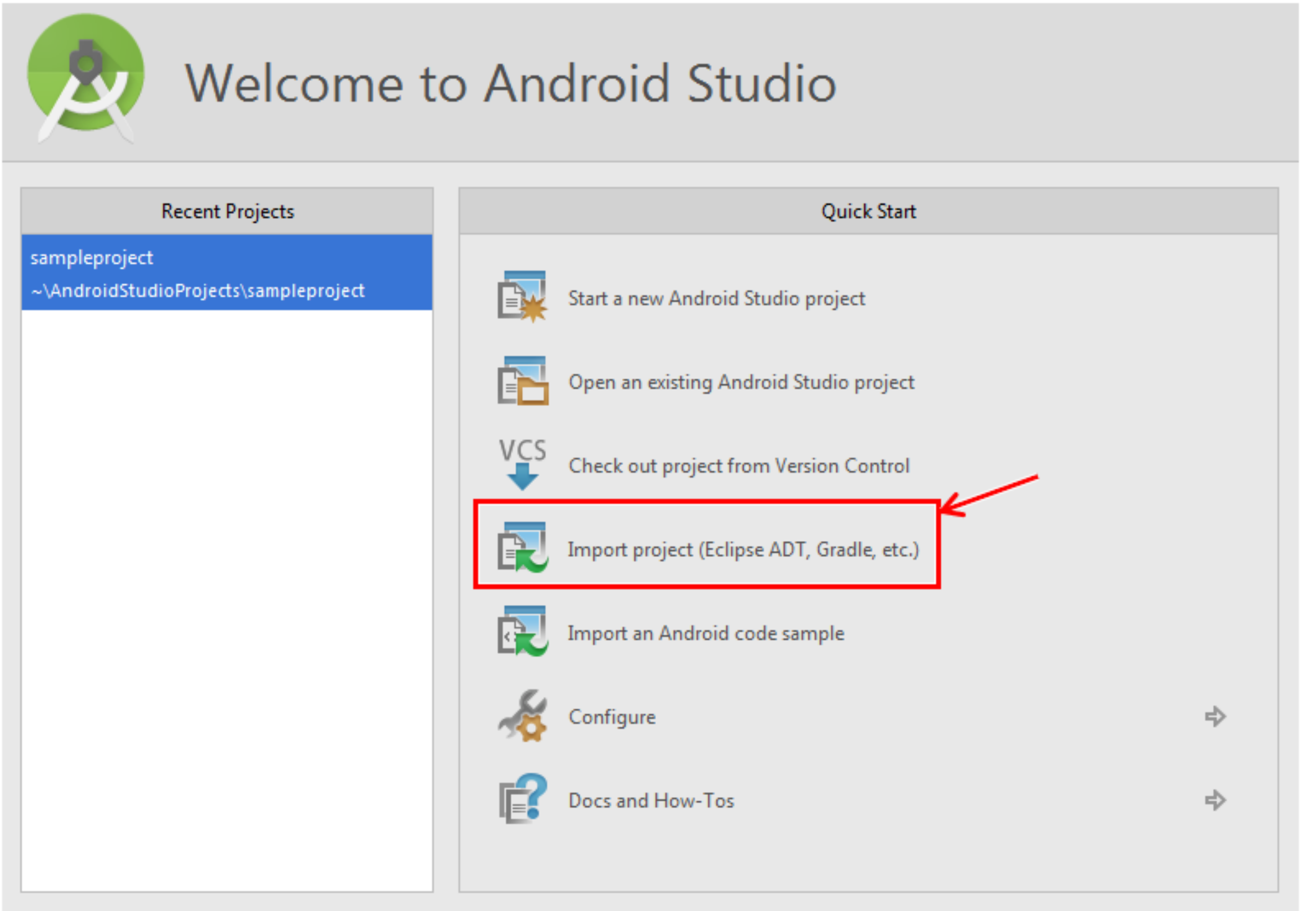
マイグレーションに関するより詳細については、Googleから提供されるガイドドキュメントを参照してください。
https://developer.android.com/sdk/installing/migrate.html

マイグレーション手順

EclipseのADTプラグインで開発していたプロジェクトを、Android Studioにマイグレーションすることができます。マイグレーションは、下記の順に進みます。

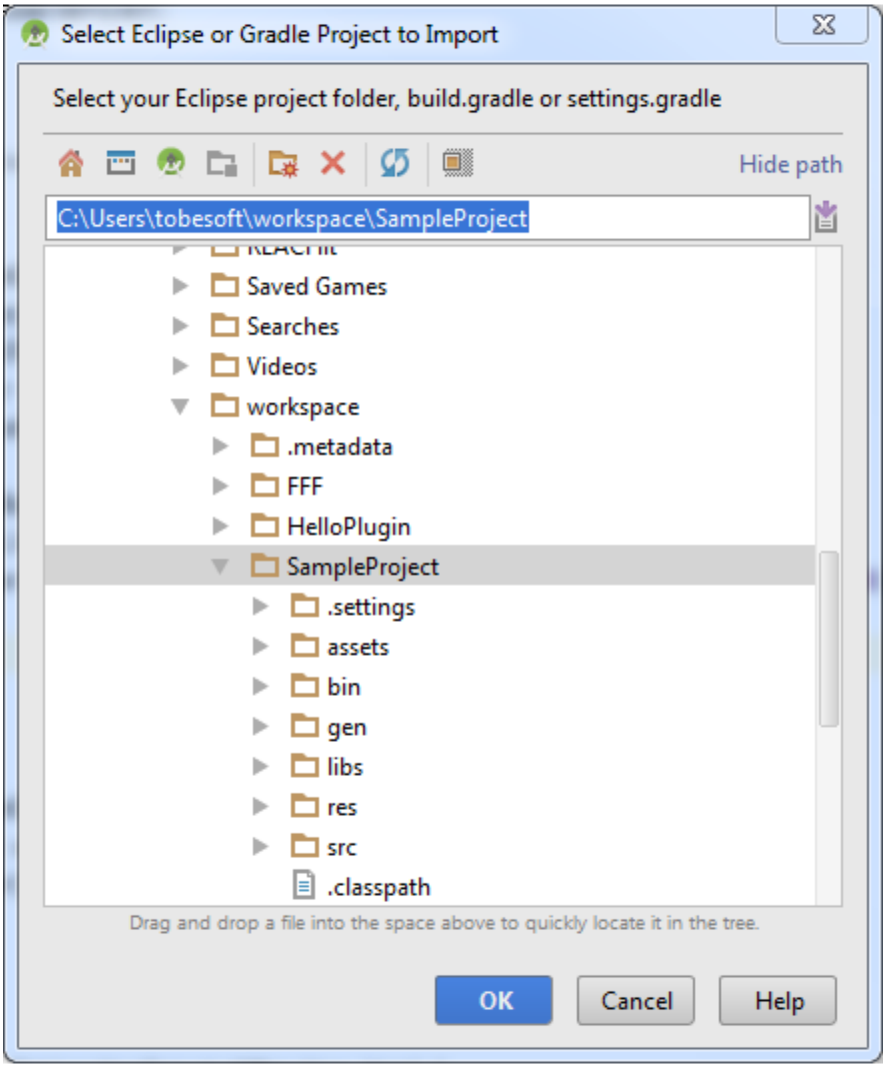
1. [Import project] 機能を実行します。

   既存に開いたプロジェクトがない場合、クイックスタート（Quick Start）ウィンドウで新しいプロジェクトを作成します。下の図で「Import Project (Eclipse ADT, Gradle, etc.)」を選択します。他のプロジェクトの作業中に、新しいプロジェクトを作成しようとすると「File > New > Import Project」項目を選択して、機能を実行することができます。



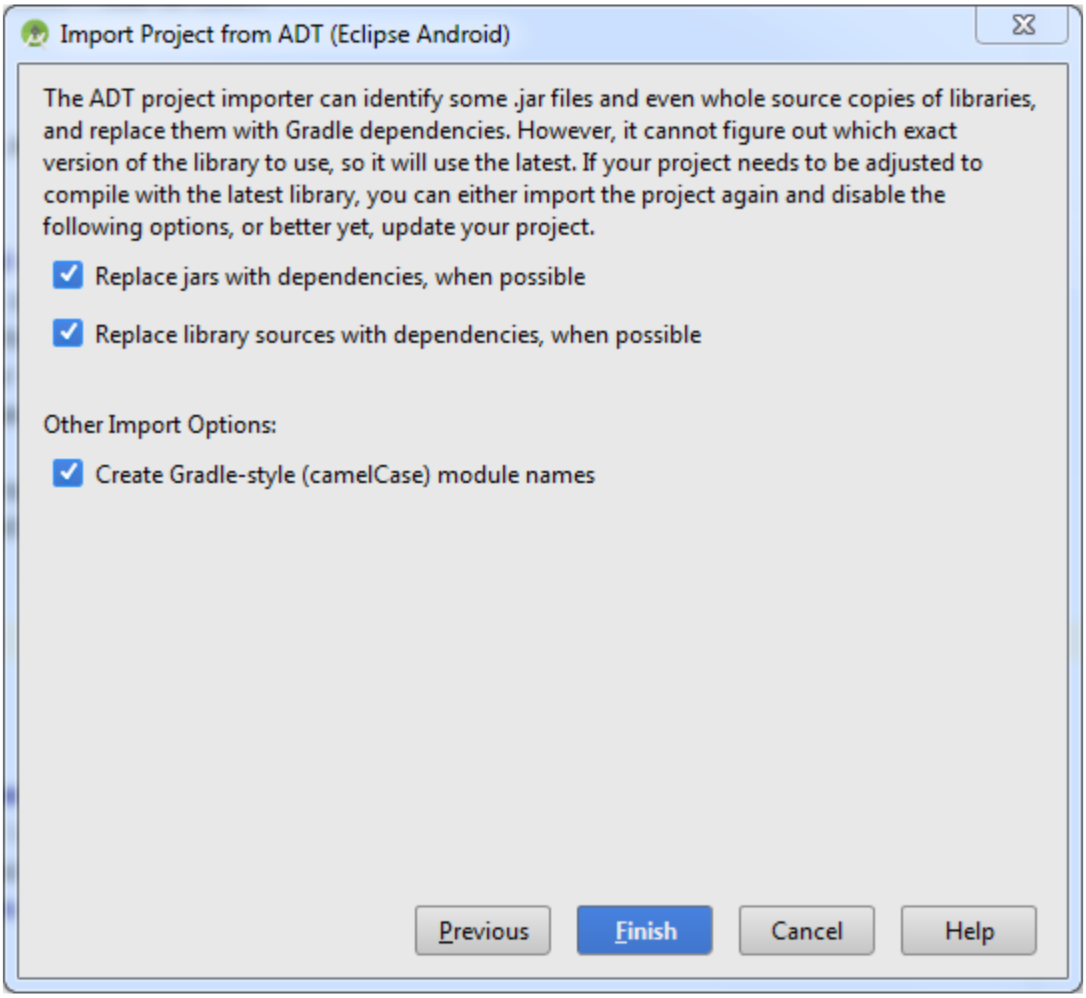
2. インポートするプロジェクトを選択します。

Eclipseの workspaceフォルダの下で、既存に作成したプロジェクトフォルダを選択します。



3. パスとオプションを選択して、「Finish」ボタンをクリックします。

同じアプリケーション名でパスを指定します。同じ名前を持つプロジェクトがある場合には、別の名前で変更することができます。



build.gradle

マイグレーション作業後のビルドが正常に処理されない場合があります。このような場合には、build.gradleの設定をいくつか変更する必要があります。よくエラーが発生る部分は、SDKのバージョンが不適切に設定される場合もあり、dependencies項目のバージョンが不適切な場合があります。

> 次のコードは例示として提供するものであり、実際のオプションは開発環境によって異なる場合があります。

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 22
    buildToolsVersion "23.0.3"
    defaultConfig {
        applicationId "com.tobesoft.sampleproject"
        minSdkVersion 15
```

```
        targetSdkVersion 22
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), '
proguard-rules.pro'
        }
    }
}
dependencies {
    compile 'com.android.support:appcompat-v7:22.2.1'
    compile 'com.android.support:support-v4.20.0.0'
    compile 'com.google.android.gms:play-services:+'
    compile files('libs/nexacro14.jar')
}
```

全オプションに関する説明は、Googleが提供する文書を参照してください。
http://developer.android.com/tools/building/configuring-gradle.html

AndroidManifest.xml

Googleプレイのサービス関連ライブラリの設定が既存の Eclipse ADTプラグインの開発環境と変わった部分があります。AndroidManifest.xmlファイルはそのまま使用できますが、エラーがある場合は、そのファイル内の下のオプションを変更します。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tobesoft.sampleproject"
    android:versionCode="1"
    android:versionName="1.0" android:installLocation="auto">
...
    <meta-data android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />
    </application>
</manifest>
```

> Googleプレイのサービス関連ライブラリは、バージョン5から使用することができますが、Android Studioは、7.5.0から gradleで dependencies処理されます。そこでAndroidManifest.xmlの設定では、SDKの play_serviceバージョンを持ってくるように設定します。

注意

> nexacro platformで提供する APIの Mapオブジェクトがマイグレーション後に正常に動作しない可能性があります。その問題は検討中です。Mapオブジェクトを使用しているサービスがある場合は、マイグレーションを保留することをお勧めします。

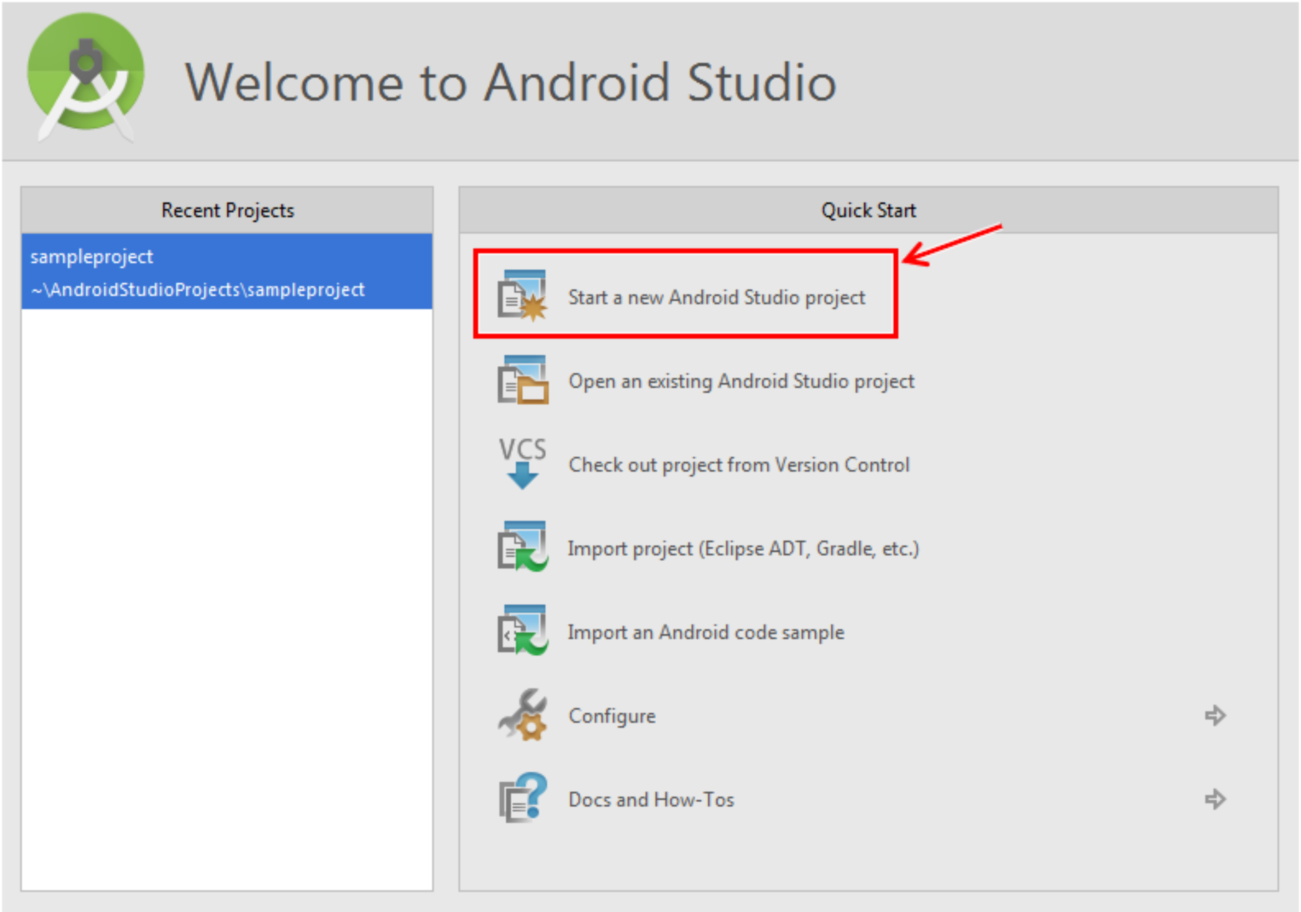
4.2 アプリプロジェクト開発

アプリ開発環境が設定された場合は、Android Studioまたは Eclipseを実行して、Android OSで動作するアプリケーションを作成することができます。アプリのプロジェクトを進行する前に、nexacro studioで開発されたアプリケーションで作成されたアーカイブファイルは、指定されたパスに移動されなる必要があります。

4.2.1 プロジェクト生成

Android Studio

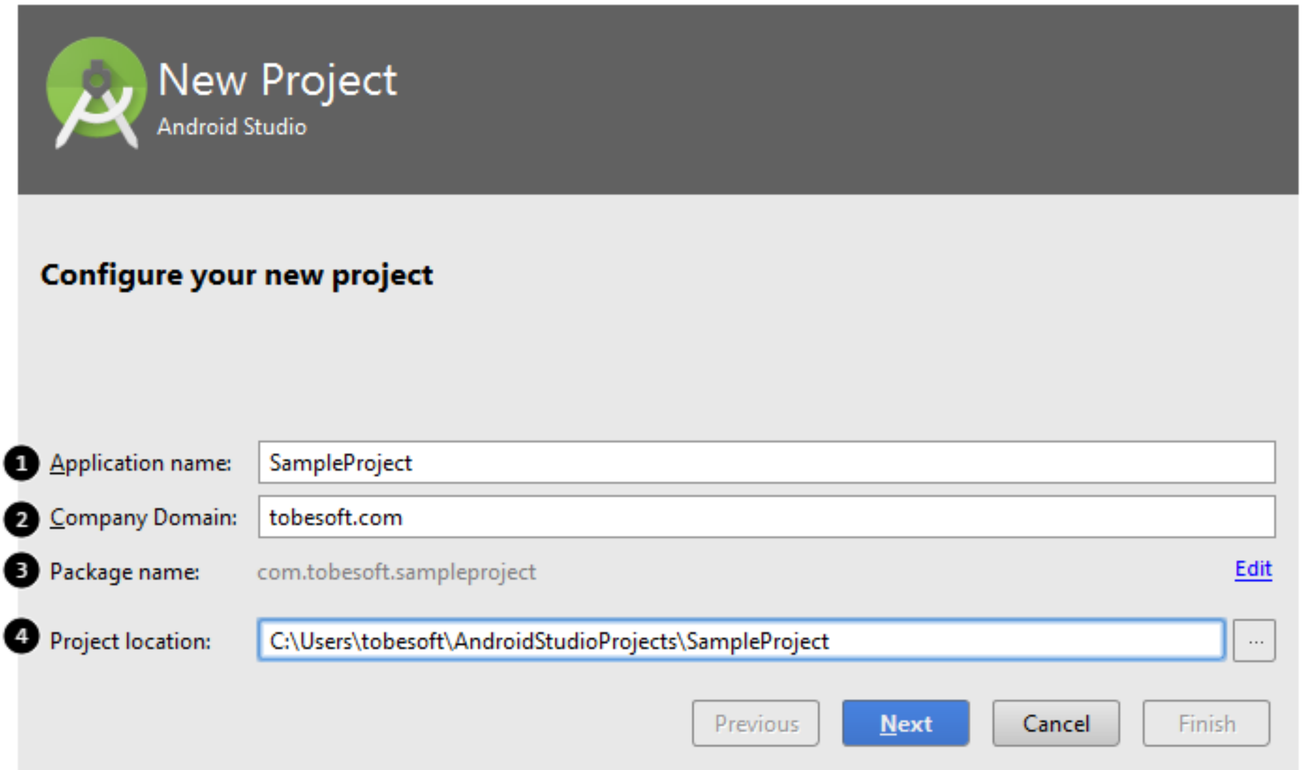
既存に開いたプロジェクトがない場合は、クイックスタート（Quick Start）ウィンドウで、新しいプロジェクトを作成します。下の画面から「Start a new Android Studio project」を選択します。



他のプロジェクトの作業中に新しいプロジェクトを作成する場合は、以下のメニューから作成することがで

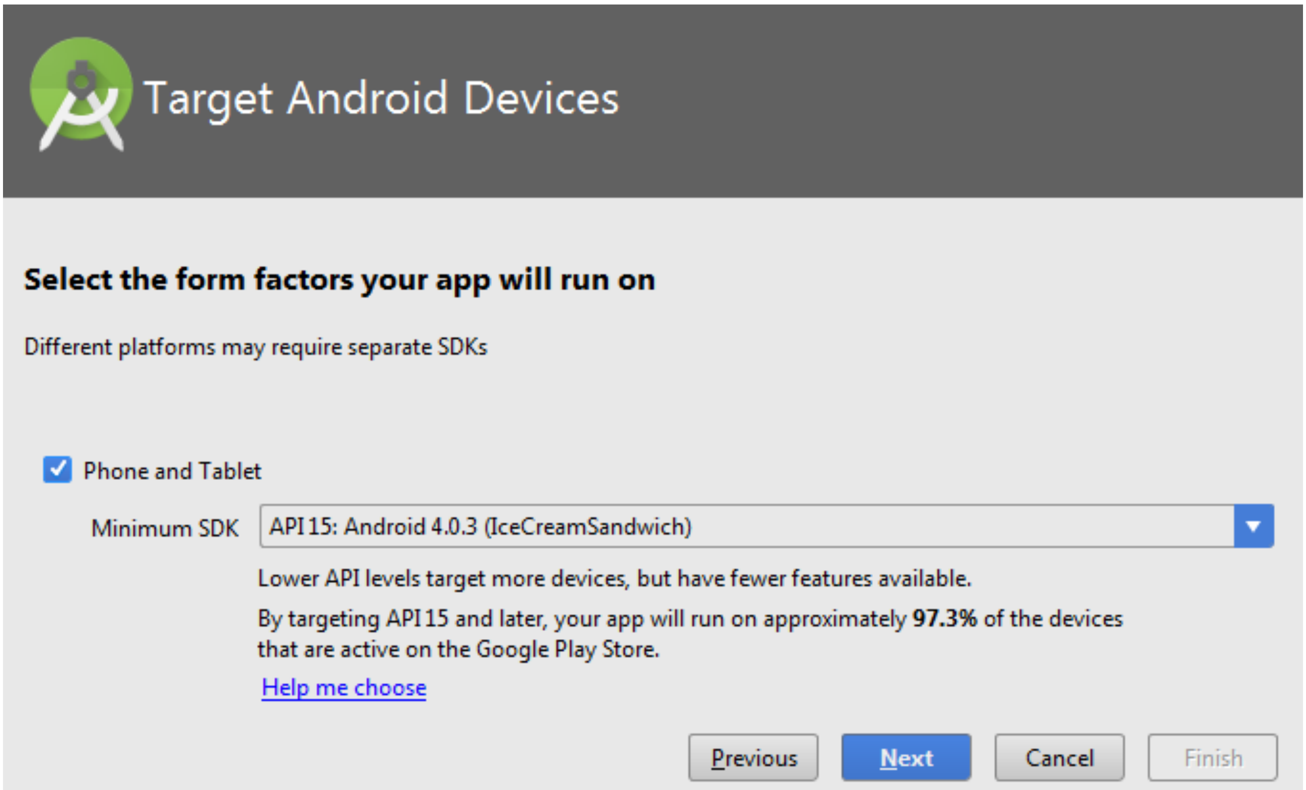
きます。

File > New > New Project

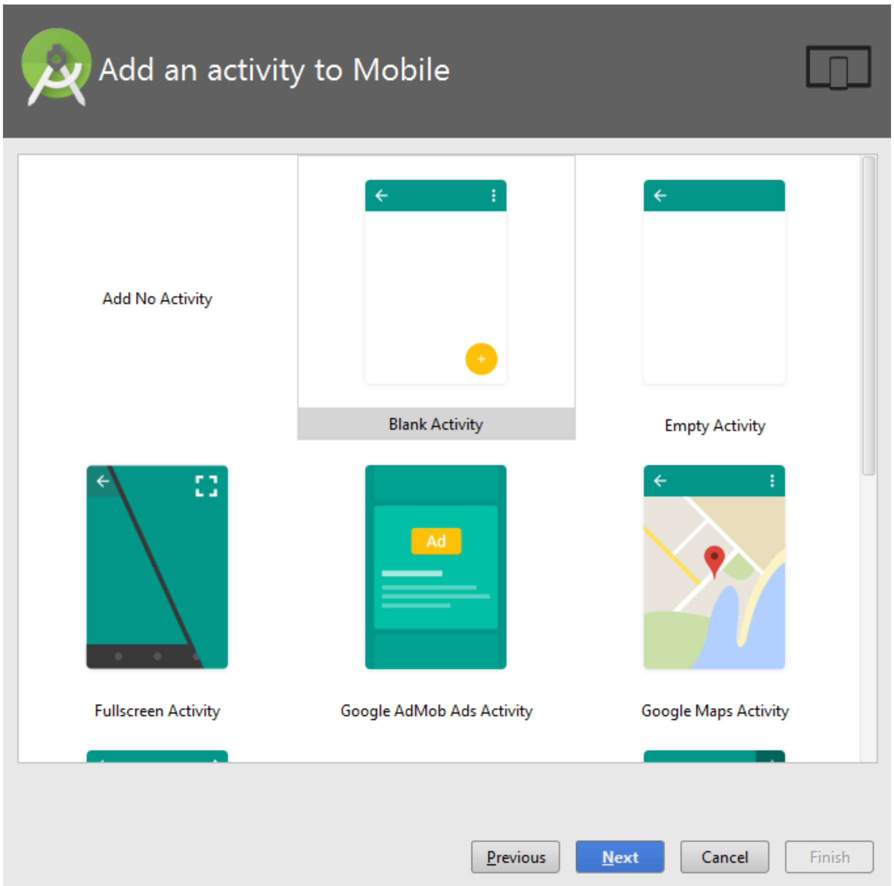


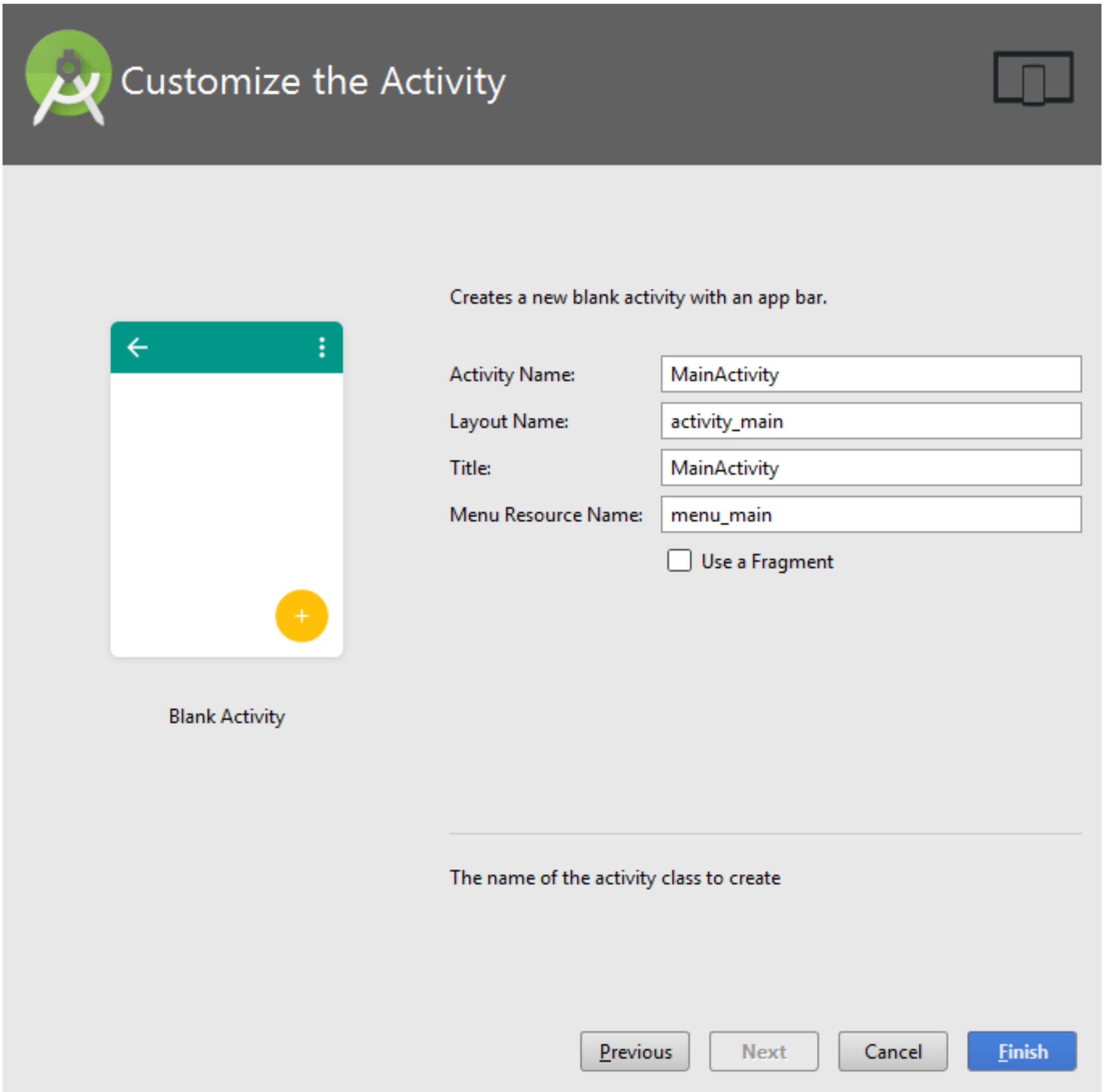
	項目	説明
❶	Application Name	生成するアプリ名を入力します。
❷	Company Domain	会社のドメイン名を入力すると、パッケージ名を自動的に生成します。
❸	Package Name	入力された会社のドメイン名に基づいてパッケージ名を生成します。
❹	Project location	プロジェクトを保存する場所を指定します。

Minimum SDKを設定します。 SDKを変更するたびにアイテムの下にどのように多くのデバイスとの互換性を提供するかについての情報を提供します。



Blank Activityを選択してから追加情報を入力して「Finish」ボタンをクリックすると、新しいプロジェクトが作成されます。





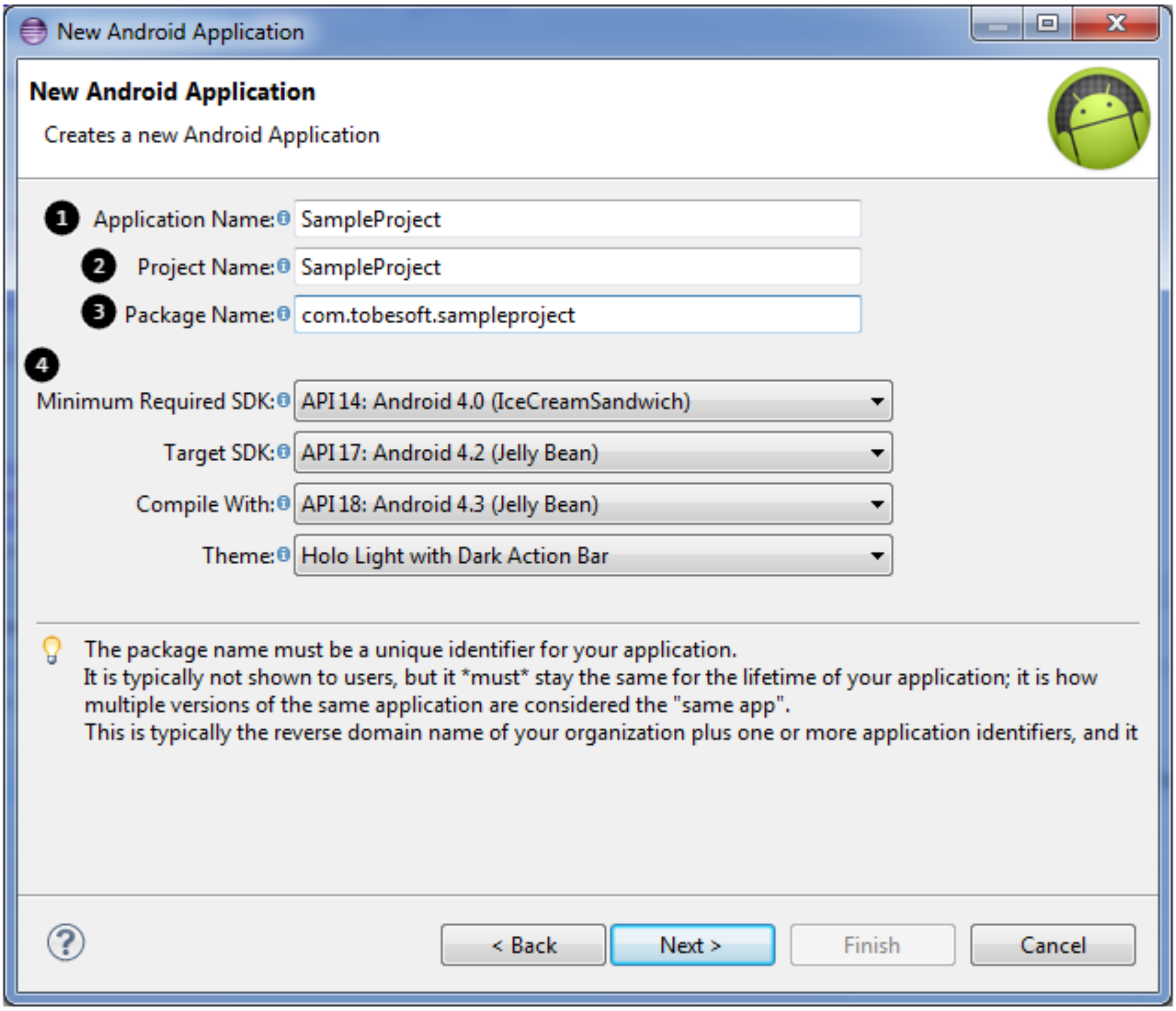
Eclipse

nexacro platformで開発されたアプリケーションを保存するAndroidプロジェクトを作成し、デフォルト環境を設定する必要があります。新しいプロジェクトは、以下のメニューから作成することができます。

```
File > New > Android Application Project
```

> 「Android Application Project」項目が表示されない場合は、ADTプラグインが正常にインストールされていないことです。該当項目をもう一度確認してください。

> プロジェクトの設定は、EclipseやAndroid SDK バージョンによって多少異なる場合があります。



	項目	説明
❶	Application Name	作成するアプリ名を入力します。
❷	Project Name	作成するプロジェクト名を入力します。プロジェクト名は、使用しているワークスペース内で固有の名前である必要があります。
❸	Package Name	ドメインの形でパッケージ名を入力します。
❹	Minimum Required SDK	nexacro platformは、Android2.3(Gingerbread)からサポートします。

> アンドロイド4.4(Kitkat)以上のSDKを使用時にMinumum Required SDKを3.2(Honeycomb)以下に設定すると、下位互換性を維持するためにappcompat_v7プロジェクトを自動生成します。
> http://developer.android.com/tools/support-library/features.html

ただし、nexacrp platformアプリケーションの作成時には影響を受けないため、プロジェクトを作成しないことをお勧めします。このような現象を回避するにはMinumum Required SDKを4.0(IceCreamSandwich)に設定し、プロジェクトを作成した後、AndroidManifest.xmlの設定時に使用するサブバージョンを再指定することができます。

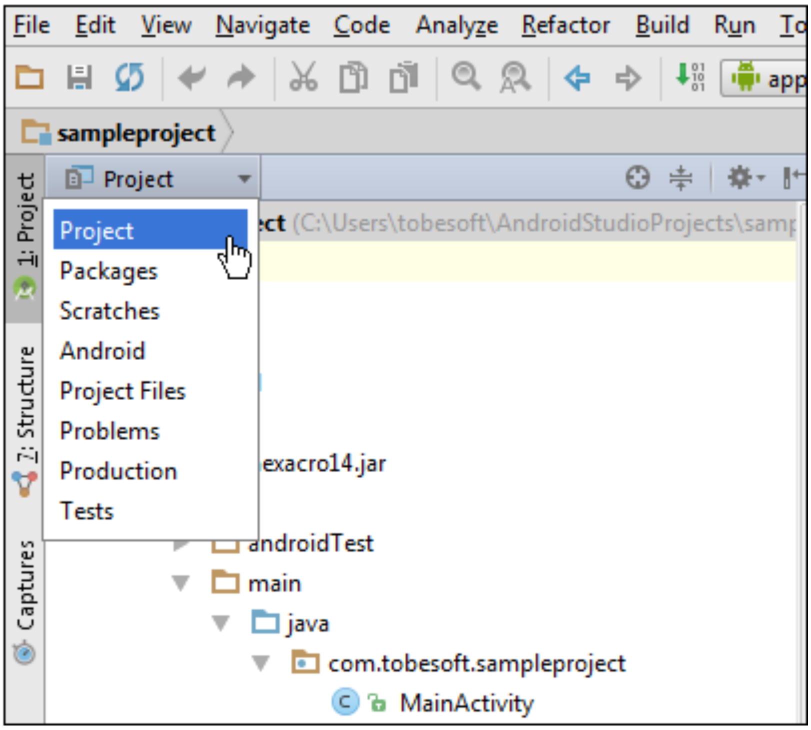
4.2.2 nexacro platformライブラリ設定

Androidのプロジェクトからnexacro platformに最適化された環境を作成するために追加で提供されるnexacro platformライブラリファイルを設定する必要があります。

> libnexacro14.soファイルは、C/ C++言語で開発されたライブラリーを意味します。

Android Studio

まず、プロジェクトウインドウの上部でビューオプションを「Project」に変更します。デフォルト値は「Android」です。

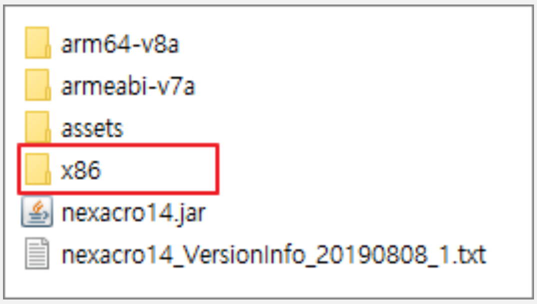


予め提供されたライブラリファイルをプロジェクトフォルダに含まれている libsフォルダにコピーします。nexacro14.jarファイルは、「プロジェクト > app > libs」フォルダの直下にコピーして、libnexacro14.soファイルは、下記を参照して「プロジェクト > app > src > main」フォルダの下にコピーします。生成されたフォルダ構造は以下の図のとおりです。

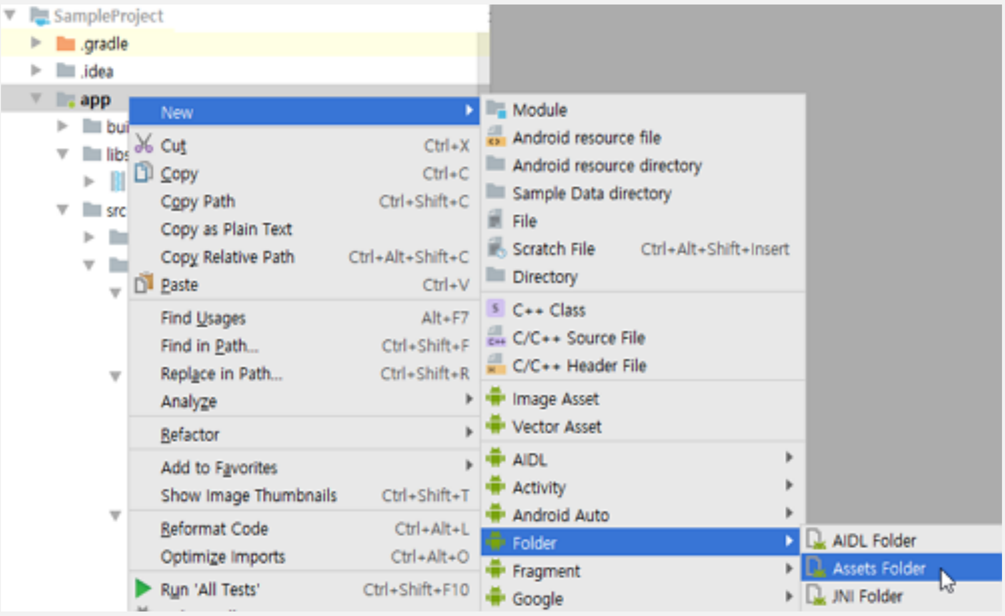
ファイル	パス
Javaアーカイブ（nexacro14.jar）	[プロジェクトパス]\app\libs\

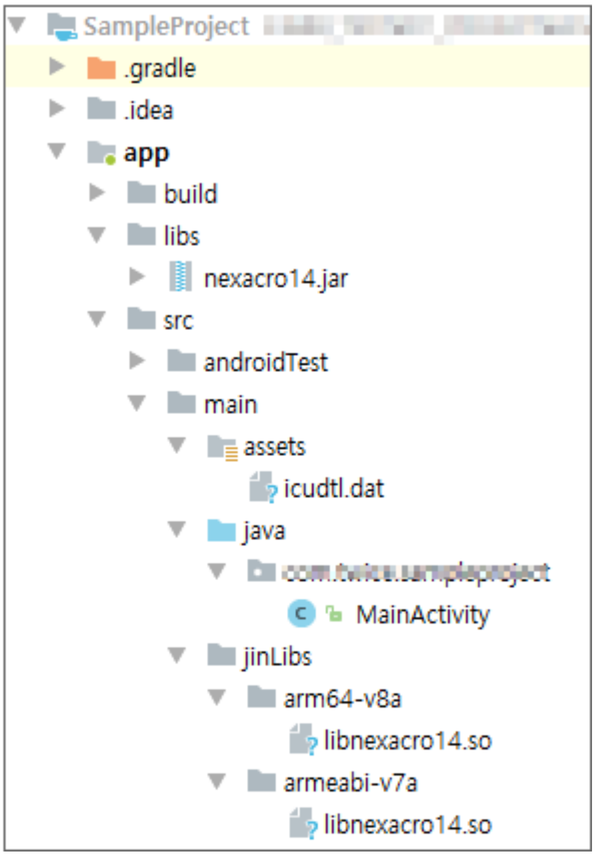
ファイル	パス
32ビット共有ライブラリ（libnexacro14.so）	[プロジェクトパス]\app\src\main\jniLibs\armeabi-v7a\
64ビット共有ライブラリ（libnexacro14.so）	[プロジェクトパス]\app\src\main\jniLibs\arm64-v8a\
リソースファイル（icudtl.dat）	[プロジェクトパス]\app\src\main\assets\

配布ファイルのx86フォルダ内のファイルは、特定のIntel CPUが搭載されているデバイスで使用するためのアプリを開発する際に使用する共有ライブラリファイルです。該当のメーカーでx86サポートが必要な場合にのみ注意して使用してください。



assetsフォルダはappを選択し、コンテキストメニューから[New > Folder > Assets Folder]項目を選択して追加します。



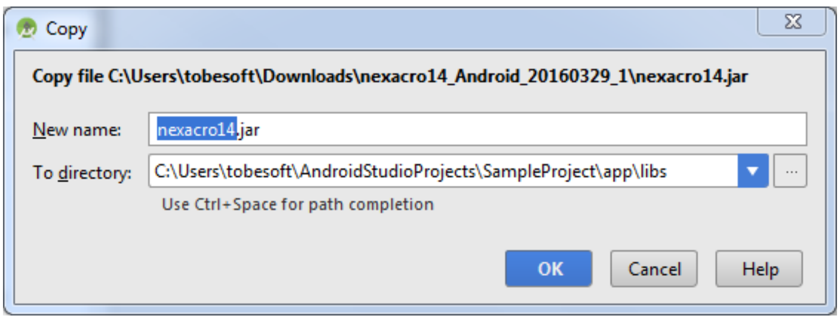


Googleのポリシーにより、2019年8月1日以降Google Playで公開するアプリは64ビットアーキテクチャをサポートする必要があります。
https://developer.android.com/distribute/best-practices/develop/64-bit

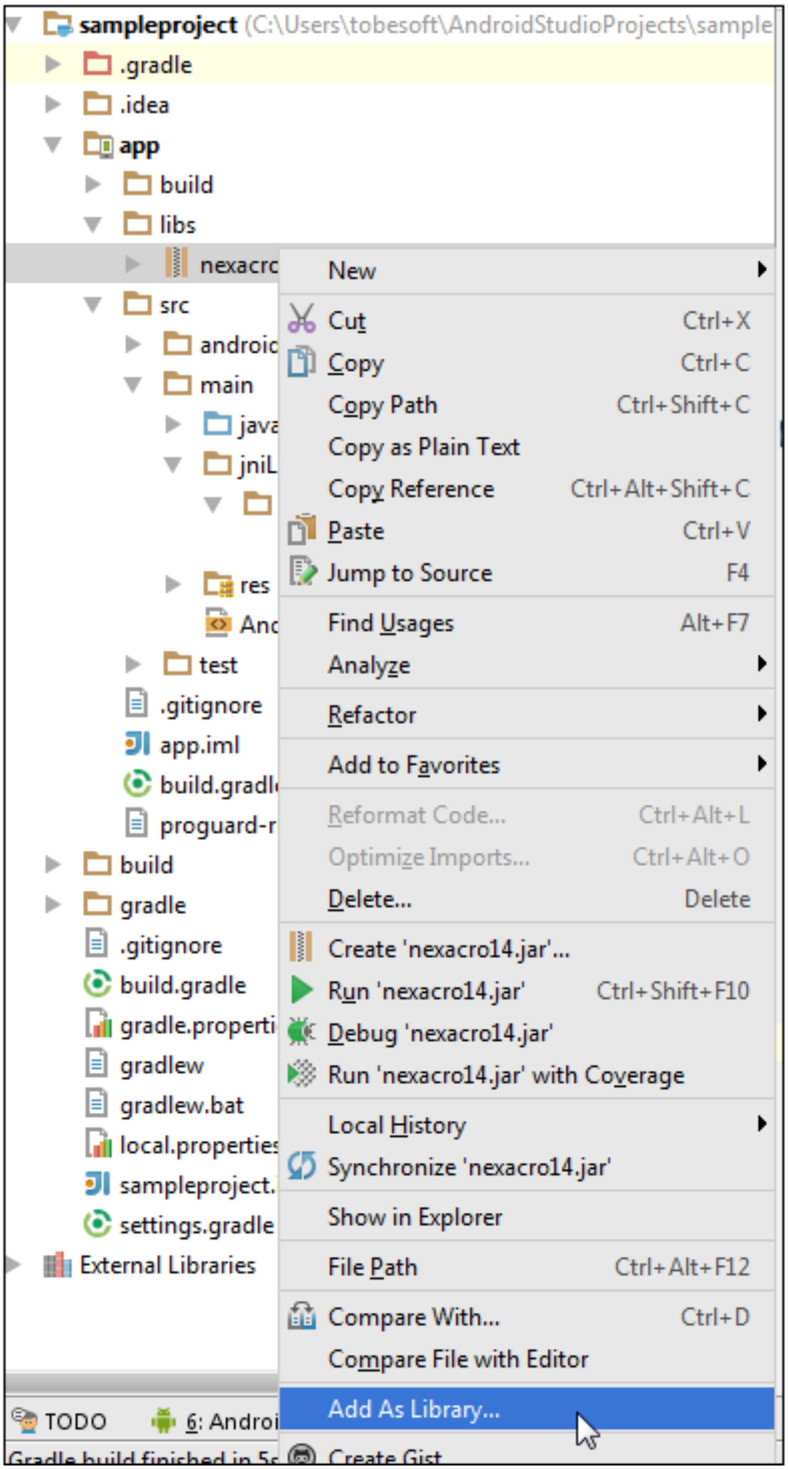
32ビットアーキテクチャのCPUを使用しているモバイルデバイスでは、64ビット用の共有ライブラリが動作しません。32ビット用と64ビット用の共有ライブラリが両方とも存在する場合には、64ビット用のライブラリを優先します。

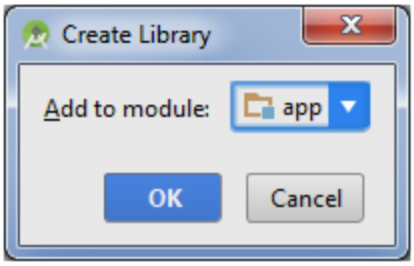
64ビットのアプリはAndroid 5.0以上の環境で実行することができます。

ファイルを追加するときに、次の図のようにファイル名とパスを変更できるウィンドウが表示されます。基本設定された値で「OK」ボタンをクリックします。

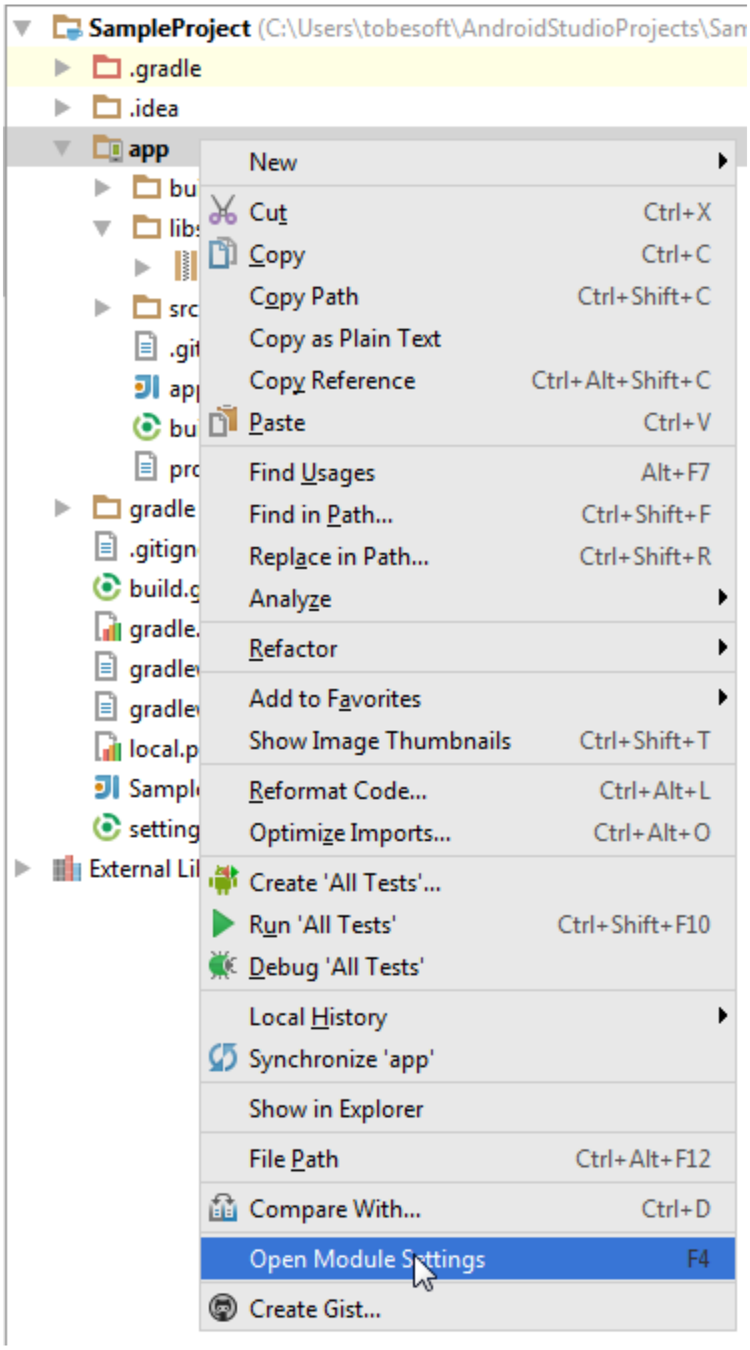


追加された nexacro platformライブラリをプロジェクトから参照するようにライブラリに追加します。nexacro14.jarファイルを選択し、コンテキストメニューから「Add As Library」項目を選択します。

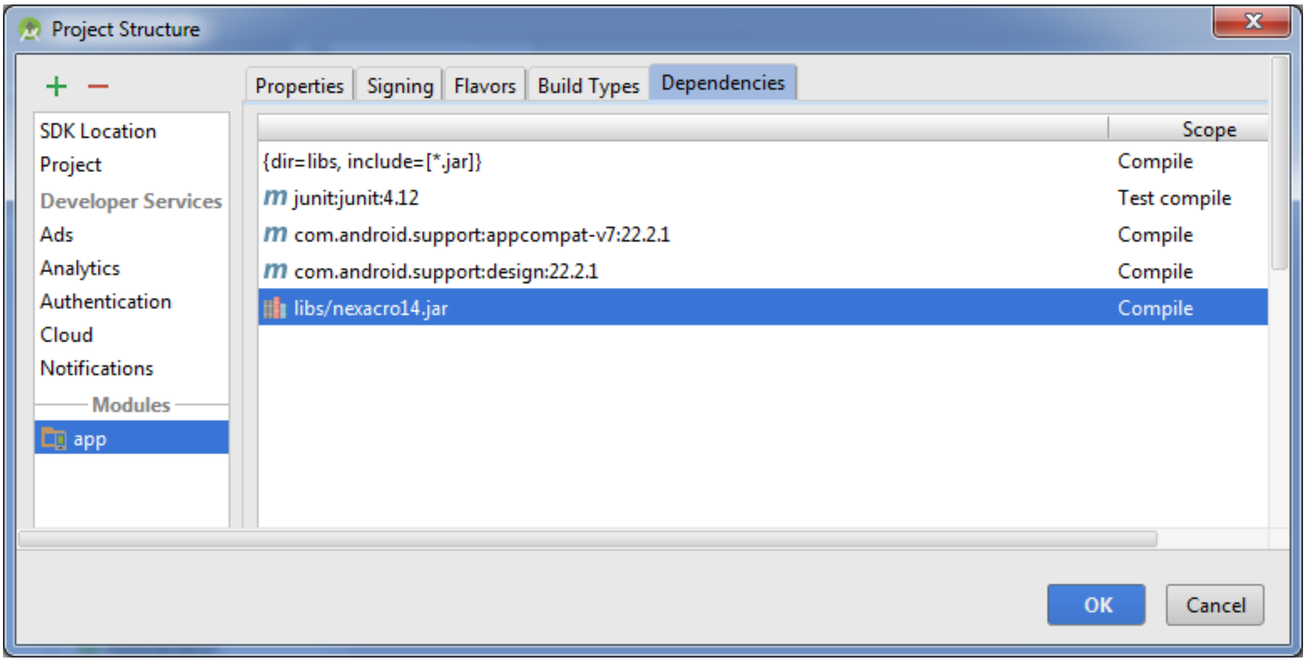




ライブラリが正常に追加されたことを確認するには、「プロジェクト > app」項目を選択し、コンテキストメニューから「Open Module Settings」項目を選択します。



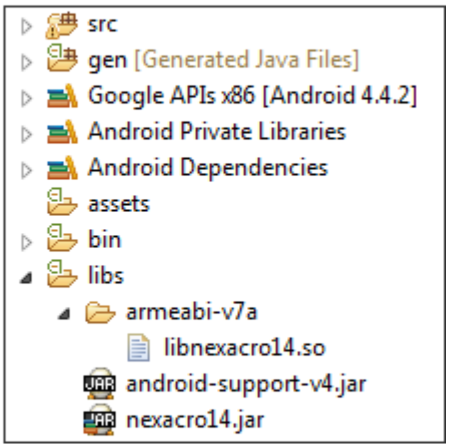
「Modules > app」項目で上部のタブを「Dependencies」に変更すると、ライブラリが追加されたことを確認することができます。



Eclipse

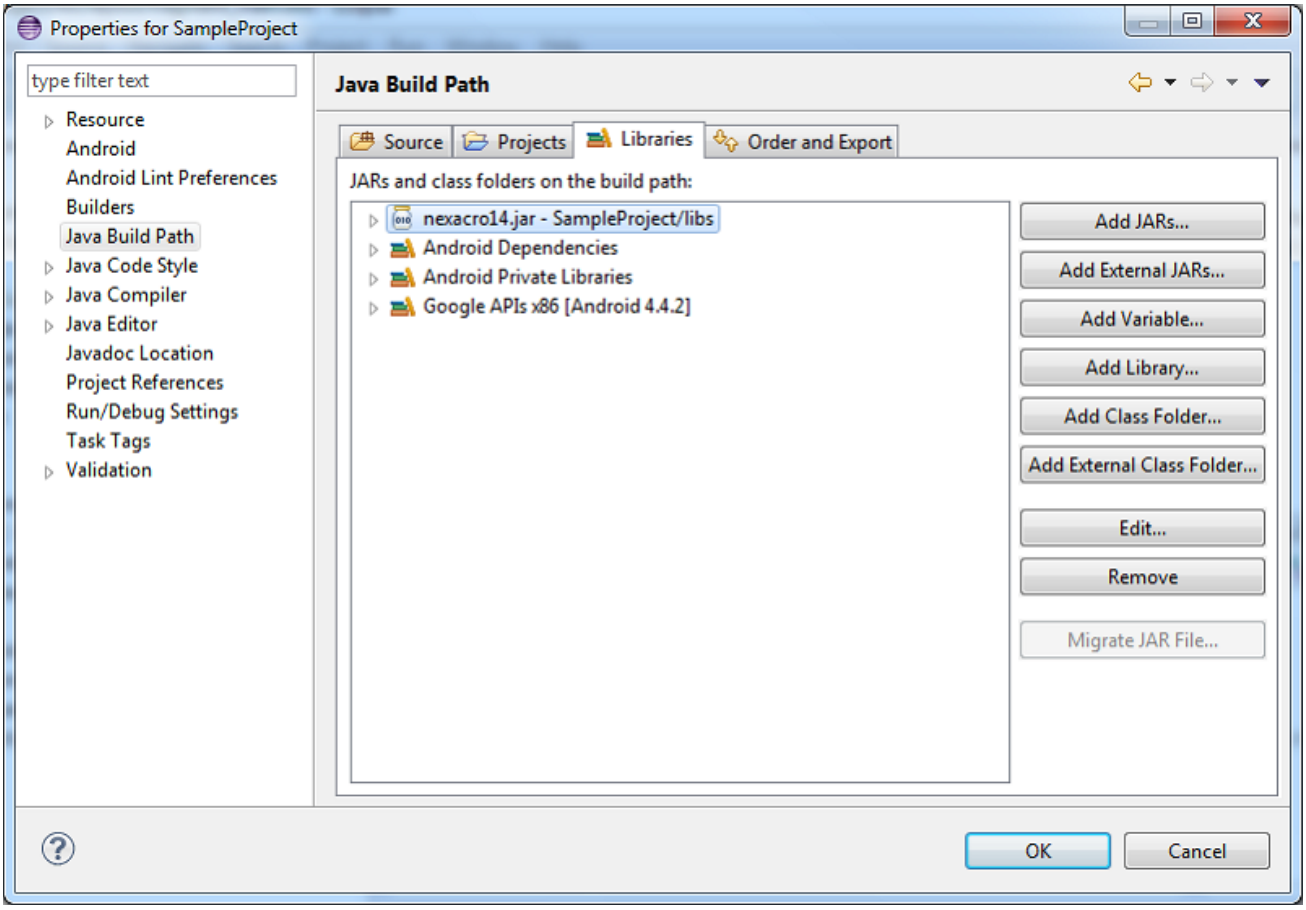
> nexacro platform 14.0.1.3000以降のバージョンからFCMサービス機能が追加され、Eclipseをサポートしなくなりました。Android Studioでのみアプリプロジェクトをビルドすることができます。
> Eclipseでは64ビット用のライブラリもサポートしません。
> ただし、既存のユーザーのために本章の説明は維持します。

予め提供されたライブラリファイルをプロジェクトフォルダに含まれているlibsフォルダにコピーします。nexacro14.jarファイルは、libsフォルダの直下にコピーし、libnexacro14.soファイルは armeabi-v7aフォルダを新規作成してそこにコピーします。生成されたフォルダ構造は以下の図のとおりです。



追加されたnexacro platformライブラリをプロジェクトから参照するようにJava Build Pathを設定します。下のメニューに移動して、「Add Jars」ボタンを選択し、libsフォルダ内のnexacro14.jarファイルを選択します。

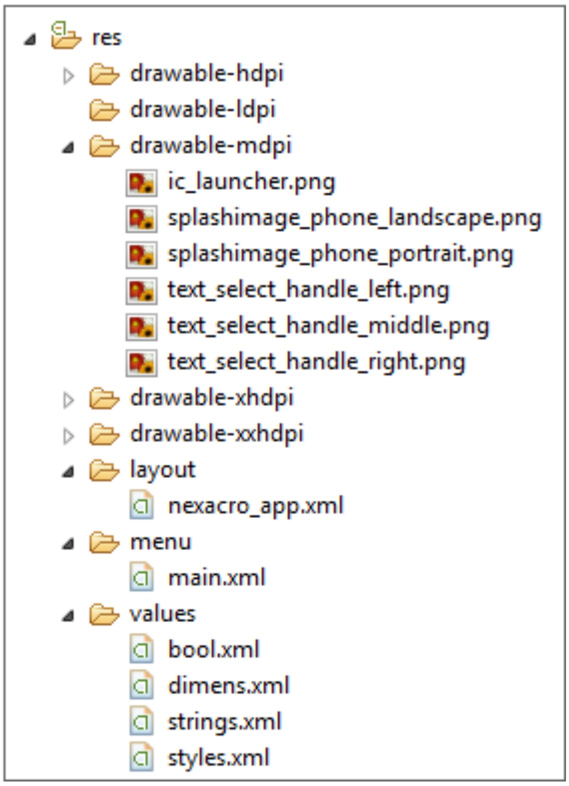
```
File > Properties > Java Build Path > Libraries > Add JARs
```



4.2.3　リソース設定

アプリで使用するローディングイメージ、アイコン、メッセージ、レイアウトなどを設定するための手順です。進行するプロジェクトによって変更して適用することができます。

各リソースを設定すると、以下のようなフォルダ構造を持ちます。



イメージ設定

ローディングイメージ、アプリアイコンをresフォルダの下(drawable-mdpiフォルダ)に指定します。適用するイメージファイルは、アプリがサポートする解像度によって設定することができます。詳細は、Androidのマニュアルをご参照ください。

http://developer.android.com/guide/practices/screens_support.html

項目	説明
icon.png	アプリケーションのデスクトップにインストールされるアイコンイメージです。 AndroidManifest.xmlファイルで指定されたファイル名を変更できます。

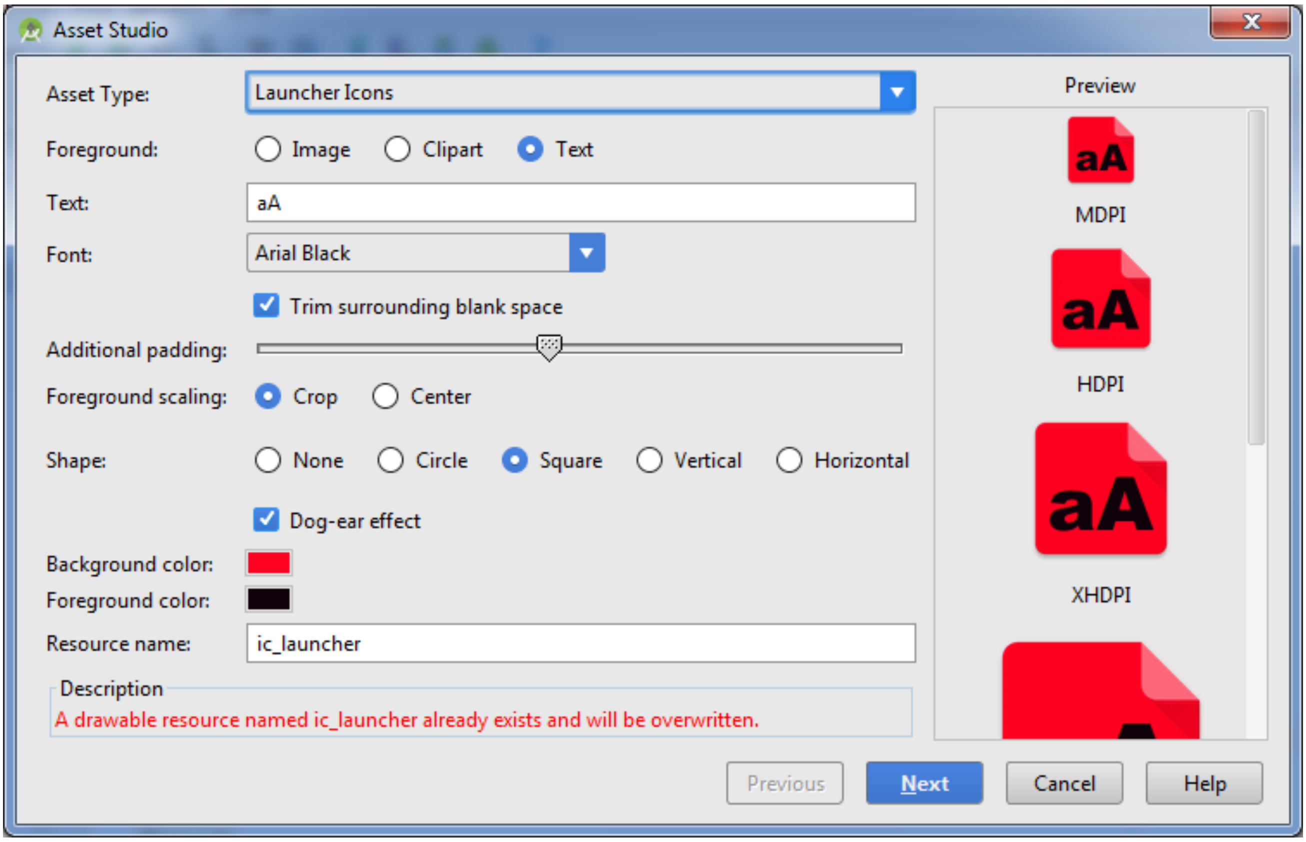
項目	説明
splashimage_phone_landscape.png	Androidフォンの横方向のスプラッシュイメージです。(すべて小文字)
splashimage_phone_portrait.png	Androidフォンの縦方向のスプラッシュイメージです。(すべて小文字)
splashimage_pad_landscape.png	Androidタブレットの横方向のスプラッシュイメージです。(すべて小文字)
splashimage_pad_portrait.png	Androidタブレットの縦方向のスプラッシュイメージです。(すべて小文字)

ファイル名が「text_select_handle~」で始まるファイルは、テキストの選択時に選択範囲を処理するために使用するイメージです。現在のバージョンでは、該当イメージを直接指定する必要があります。

アイコンファイルは、プロジェクトの作成後に変更できます。

イメージ設定 | Android Studio

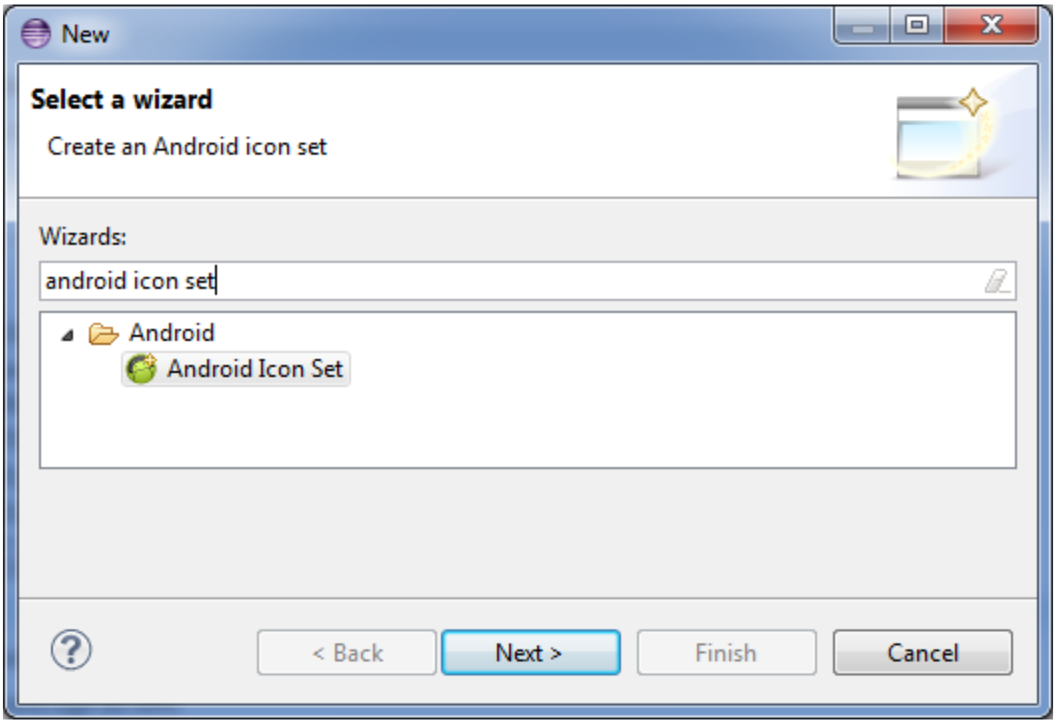
「プロジェクト > app」項目を選択した後、コンテキストメニューから「New > Image Asset」項目を選択した後に表示された「Asset Studio」でアイコンファイル関連の設定を変更することができます。



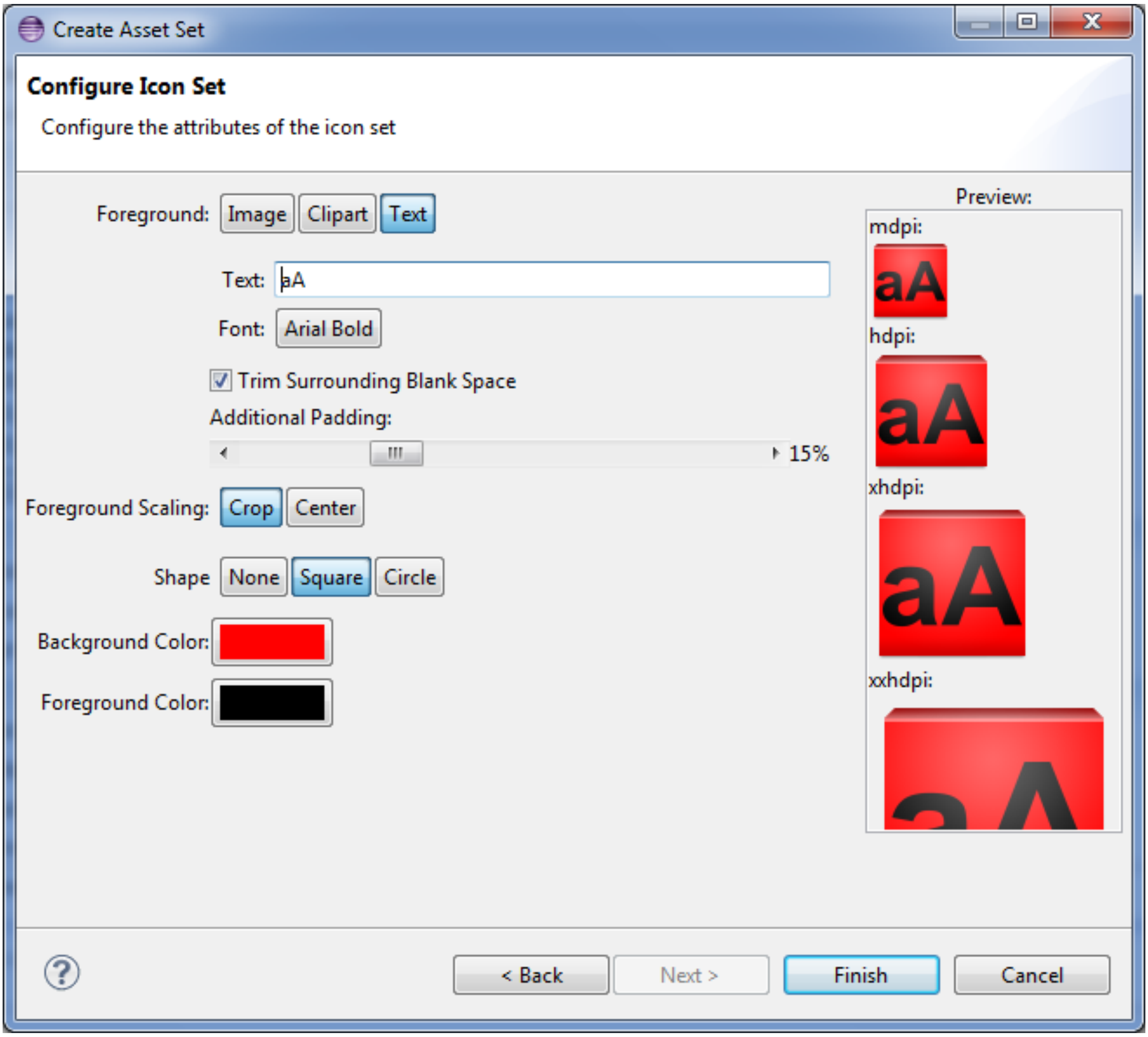
http://developer.android.com/intl/ko/tools/help/image-asset-studio.html

イメージ設定 | Eclipse

プロジェクトを選択した後、コンテキストメニューから「New > Other」項目を選択した後に表示されたウィザードウィンドウで「android icon set」で検索して、「Android Icon Set」項目を選択します。



「Configure Icon Set」項目で、アイコンファイル関連の設定を変更することができます。



メッセージ設定

アプリの実行時に使用するメッセージを設定します。プロジェクトの作成時に作成されたresフォルダの下にvaluesフォルダのstrings.xmlファイルを以下のように変更します。

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Nexacro 14</string>
    <string name="action_settings">Settings</string>

    <!-- Clipboard -->
    <string name="copy">Copy</string>
```

```
    <string name="paste">Paste</string>
    <string name="cut">Cut</string>
    <string name="close">Close</string>
    <string name="selectall">Select All</string>
    <string name="selectword">Select Word</string>

    <string name="needupdate">It is need to update. \r\nAfter completing,
Please restart.</string>
    <string name="loadingFail">First loading is fail. \r\nPlease restart.</
string>
    <string name="updateFail">Update is fail. \r\nPlease restart.</string>
    <string name="notexist">Start file is NOT exist. \r\nPlease restart.</
string>
    <string name="BeingUpdated">Being updated.</string>
    <string name="wantreplace">There is a file of the save name. Do you want to
replace? </string>
    <string name="ok">OK</string>
    <string name="cancel">Cancel</string>
    <string name="move">Move</string>
    <string name="upper">Upper</string>
    <string name="filter">Filter</string>
    <string name="home">Home</string>
    <string name="nofilename">No File Name.</string>
    <string name="checkforupdates">Check for updates.</string>
    <string name="installforupdates">Install for updates.</string>
    <string name="downloadingforupdates">Downloading for updates.</string>
    <string name="force_close">Update is Completed.</string>
    <string name="LoadingPleaseWait">Loading… \r\nPlease Wait…</string>

    <!-- Accessibility widget role -->
    <string name="alert">alert</string>
    <string name="pushbutton">pushbutton</string>
    <string name="combobox">combobox</string>
    <string name="checkbutton">checkbutton</string>
    <string name="radiobutton">radiobutton</string>
    <string name="text">text</string>
    <string name="list">list</string>
    <string name="listitem">listitem</string>
    <string name="statictext"></string>
    <string name="graphic">image</string>
    <string name="spacebar">space bar</string>
```

```
    <string name="textdeleted">deleted</string>

    <!-- Accessibility widget statue -->
    <string name="checked">checked</string>
    <string name="unchecked">unchecked</string>

    <!-- Accessibility reaction -->
    <string name="click">click</string>

    <!-- Splash message  -->
    <string name="start_initialize">Start Initialize</string>
    <string name="check_license">Check License</string>
    <string name="load_frameworkmodule">Load Framework Module</string>
    <string name="load_extendmodule">Load Extend Module</string>
    <string name="execute_frameworkscript">Execute Framework Script</string>
    <string name="load_application">Load Application</string>
    <string name="load_failapplication">Load Fail Application Initialize. Click
And Close</string>

</resources>
```

英語メッセージのほか、日本語メッセージを追加するには、resフォルダの下にvalues-jaのようなフォルダを追加し、その中にstrings.xmlファイルを追加します。

> app_name項目は、実際のデバイスにインストールされるアイコン名を定義しています。該当項目を変更すると、画面に表示されるアイコン名を変更できます。

bool.xmlファイルを追加します。

bool.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
    <bool name="is_jellybean_or_higher">false</bool>
</resources>
```

レイアウト設定

アプリが端末に表示されるレイアウトを指定します。 nexacro platformアプリケーションを包むための用途

であるため、デフォルト状態に設定します。

resフォルダの下にあるlayoutフォルダにnexacro_app.xmlファイルを新規作成して追加します。

nexacro_app.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- nexacro_app.xml -->
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nexacro_activity"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/transparent"
    tools:context=".deviceAPI.FileDialogActivity" >
    <com.nexacro.view.NexacroLayout
        android:id="@+id/nexacro_layout"
        android:background="@android:color/transparent"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </com.nexacro.view.NexacroLayout>
</FrameLayout>
```

4.2.4 ビルド環境設定

AndroidプロジェクトをN作成し、デフォルトで作られたMainActivity.javaファイルとAndroidManifest.xmlファイルをnexacro platform環境に合わせて変更します。

MainActivity.java

プロジェクト作成時にアクティビティ名を指定してくれますが、別に変更しない限りは、MainActivity.javaという名前で生成されます。srcフォルダの下に指定したパッケージ名の下にあるMainActivity.javaファイルを選択し、以下のように変更します。

以下の内容の中で太字で表示されたsetBootstrapURLに含まれたパラメータは、nexacro studioで作成したstart_android.jsonファイルが配置されたサーバーURLを指定し、setProjectURLメソッドに含まれたパラメータは該当パスを「/」まで指定します。

MainActivity.java

```
package com.tobesoft.sampleproject;
import com.nexacro.NexacroUpdatorActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
public class MainActivity extends NexacroUpdatorActivity {
    public MainActivity() {
        super();
        setBootstrapURL("http://172.10.10.10:8080/nexacro/start_android.json");
        setProjectURL("http://172.10.10.10:8080/nexacro/");
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        Intent intent = getIntent();
        if(intent != null) {
            String bootstrapURL = intent.getStringExtra("bootstrapURL");
            String projectUrl = intent.getStringExtra("projectUrl");
            if(bootstrapURL != null) {
                setBootstrapURL(bootstrapURL);
                setProjectURL(projectUrl);
            }
        }
```

```
        super.onCreate(savedInstanceState);
    }

    @Override
    public void setContentView(View view) {
        super.setContentView(view);
    }
}
```

プロジェクトのアーカイブファイルを作成せずに直接Webサーバーにアクセスする場合は、次のようにxadl.jsファイルを指定したsetStartupURLメソッドを追加します。この時、start_android.jsonファイルには、engine項目のみ設定します。

```
super();
setBootstrapURL("http://172.10.10.10:8080/nexacro/start_android.json");
setStartupURL("http://172.10.10.10:8080/nexacro/HBDeviceTest/HBDeviceTest.xadl.js");
```

AndroidManifest.xml

AndroidManifest.xmlファイルは、プロジェクトのルートに位置し、アプリケーションの基本的な情報と、実行時に必要な権限についての情報を含んでいます。基本的に必要な情報を以下のコードを参考にして作成します。以下の内容の中で太字で示した package、MainActivityの情報は、プロジェクトの設定によって指定する必要があります。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tobesoft.sampleproject"
    android:installLocation="auto"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="9"
        android:targetSdkVersion="19" />
    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />
```

```
<!-- asset/archive機能を使用する場合は、削除することができます。そうでない場合は、
削除することができません。 -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />

    <application
        android:allowBackup="true"
        android:hasCode="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.tobesoft.sampleproject.MainActivity"
            android:configChanges="orientation|keyboardHidden|screenSize"
            android:theme="@android:style/Theme.NoTitleBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.nexacro.NexacroActivity"
            android:alwaysRetainTaskState="true"
            android:configChanges="orientation|keyboardHidden|screenSize"
            android:launchMode="singleTop"
            android:theme="@android:style/Theme.NoTitleBar"
            android:windowSoftInputMode="adjustResize" >
        </activity>

        <meta-data
            android:name="com.google.android.gms.version"
            android:value="google_play_services_version" />
    </application>
</manifest>
```

targetsdkバージョンを28以上に指定した場合にHTTP通信を維持するためには、usesCleartextTraffic属性値を「true」に指定する必要があります。

> 未設定にした場合はデフォルト値が「false」として適用され、HTTP通信が許可されません。
> 詳細については、下記のページをご参照ください。
> https://developer.android.com/guide/topics/manifest/application-element#usesCleartextTraffic
>
> <application
> android:allowBackup="true"
> **android:usesCleartextTraffic**="true"

配布するアプリの機能によって、必要な権限とその他の情報を追加で指定します。例えば、カメラ機能を使用する場合は、次のような項目が追加される必要があります。

```
...
<uses-permission android:name="android.permission.CAMERA" />
<activity android:name="com.nexacro.deviceAPI.CameraListener"
    android:screenOrientation="landscape">
    <intent-filter>
            <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<uses-feature android:name="android.hardware.camera" android:required="false"/>
<uses-feature android:name="android.hardware.camera.autofocus"
    android:required="false"/>
...
```

機能による情報の設定時に追加できる項目は以下の通りです。必要な項目のみをコピーして AndroidManifest.xmlファイルに追加します。

DeviceAPI

```
<!-- AudioPlayer, AudioRecorder -->
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<!-- Camera -->
<uses-permission android:name="android.permission.CAMERA" />
<activity android:name="com.nexacro.deviceAPI.CameraListener" android:
screenOrientation="landscape">
    <intent-filter>
        <category android:name="android.intent.category.LAUNCHER" />
```

```
    </intent-filter>
</activity>
<uses-feature android:name="android.hardware.camera" android:required="false"/>
<uses-feature android:name="android.hardware.camera.autofocus" android:required
="false"/>
<!-- Contact -->
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
<!-- ExternalAPI -->
<uses-permission android:name="android.permission.GET_TASKS" />
<!-- Geolocation, Map -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_
COMMANDS" />
<!-- Map -->
<uses-permission android:name="com.google.android.providers.gsf.permission.READ
_GSERVICES"/>
<uses-library android:name="com.google.android.maps" />
<meta-data android:name="com.google.android.maps.v2.API_KEY" android:value="
user value" />
<!-- SMS -->
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.WRITE_SMS" />
<receiver android:name="com.nexacro.deviceAPI.SmsRecv">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.provider.Telephony.WAP_PUSH_RECEIVED" />
        <data android:mimeType="application/vnd.wap.mms-message" />
    </intent-filter>
</receiver>"
<!-- Vibrator -->
<uses-permission android:name="android.permission.VIBRATE" />
<!-- FileDialog -->
<activity android:name="com.nexacro.deviceAPI.FileDialogActivity" android:
screenOrientation="sensor">
    <intent-filter>
```

```
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

グーグルマップを使用するためにはAPI KEY値を受け取って入力する必要があります。
詳しくは、以下のリンクを参照してください。
https://developers.google.com/maps/documentation/android/start#obtain_a_google_maps_api_key

GlobalAPI

```
<!-- アカウント -->
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<!-- 電話番号 -->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<!-- 電話をかける -->
<uses-permission android:name="android.permission.CALL_PHONE" />
<!-- デバイス名 -->
<uses-permission android:name="android.permission.BLUETOOTH"/>
```

asset/archive機能

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
```

cache機能

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_INTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_INTERNAL_STORAGE" />
```

Notification機能（14.0.1.3000以降のバージョン）

14.0.1.3000以降のバージョンでは、Firebase Cloud Messaging(FCM)サービスを使用します。

About FCM messages
https://firebase.google.com/docs/cloud-messaging/concept-options

FCMサービスはAndroid4.0（API14）以降で使用することができます。該当バージョンより古いバージョンのOSではFCMサービスが正常に動作しない場合があります。

```
<!-- Adding NexacroNotificationService -->
<service android:name="com.nexacro.notification.NexacroNotificationService">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT" />
    </intent-filter>
</service>
<!-- Adding NexacroActivity intent-filter -->
<activity android:name="com.nexacro.NexacroActivity"
    android:launchMode="singleTask"
    android:noHistory="false"
    android:hardwareAccelerated="true"
    android:windowSoftInputMode="adjustResize"
    android:configChanges="orientation|keyboard|keyboardHidden|screenSize|
navigation|uiMode">
    <intent-filter>
        <action android:name="OPEN_NEXACRO_ACTIVITY_EXAMPLE" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

FCMサービスを利用する場合には、build.gradle設定にて下記の項目を追加する必要があります。Android Studioで自動的に設定されない場合は手動で追加し、Firebaseコンソールで生成したgoogle-services.jsonファイルをモジュールのパスにコピーします。

AndroidプロジェクトにFirebase追加
https://firebase.google.com/docs/android/setup

```
buildscript {
    // Adding repositories
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.2.1'
        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
        // Adding google service
        classpath 'com.google.gms:google-services:4.0.1'
    }
}
allprojects {
    // Adding repositories
    repositories {
        google()
        jcenter()
    }
}
task clean(type: Delete) {
    delete rootProject.buildDir
}
dependencies {
    implementation fileTree(include: ['*.jar'], dir: 'libs')
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.android.support:support-v4:27.1.1'
    implementation 'com.google.android.gms:play-services-maps:16.0.0'
    implementation 'com.google.android.gms:play-services-location:16.0.0'
    implementation 'com.google.firebase:firebase-core:16.0.6'
    implementation 'com.google.firebase:firebase-messaging:17.3.4'
    implementation files('libs/nexacro14.jar')
}
apply plugin: 'com.google.gms.google-services'
```

Notification機能（14.0.1.2802以前のバージョン）

14.0.1.2802以前のバージョンでは、下記のように使用します。

```
<!-- Notification -->
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<receiver
    android:name="com.nexacro.notification.NexacroNotificationReceiver"
    android:permission="com.google.android.c2dm.permission.SEND" >
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        <category android:name="com.nexacro" />
    </intent-filter>
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
        <category android:name="com.nexacro" />
    </intent-filter>
</receiver>
<service android:name="com.nexacro.notification.NexacroNotificationService" />
```

4.3 ビルド

4.3.1 アプリテスト

1. Android端末をPCにUSBで接続してすぐにテストすることができます。端末を直接接続する場合には、端末のアプリケーション開発オプションの[アプリケーション > 開発 > USBデバッグ]項目を設定し、端末に合ったドライバをインストールする必要があります。

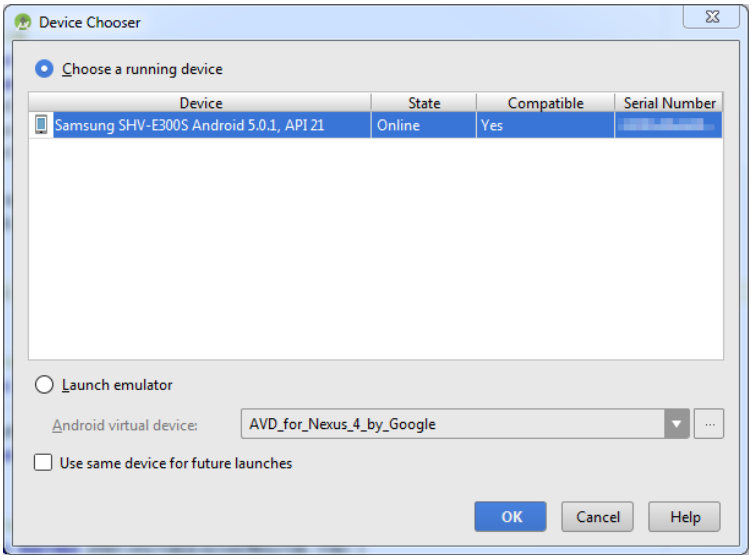
2. Android Studioまたは EclipseでRunメニューを実行します。Runメニューは、APKを作成し、端末にインストールして実行まで行うことができる機能です。実行する前に、どの端末で実行するかを選択します。

Run > Run

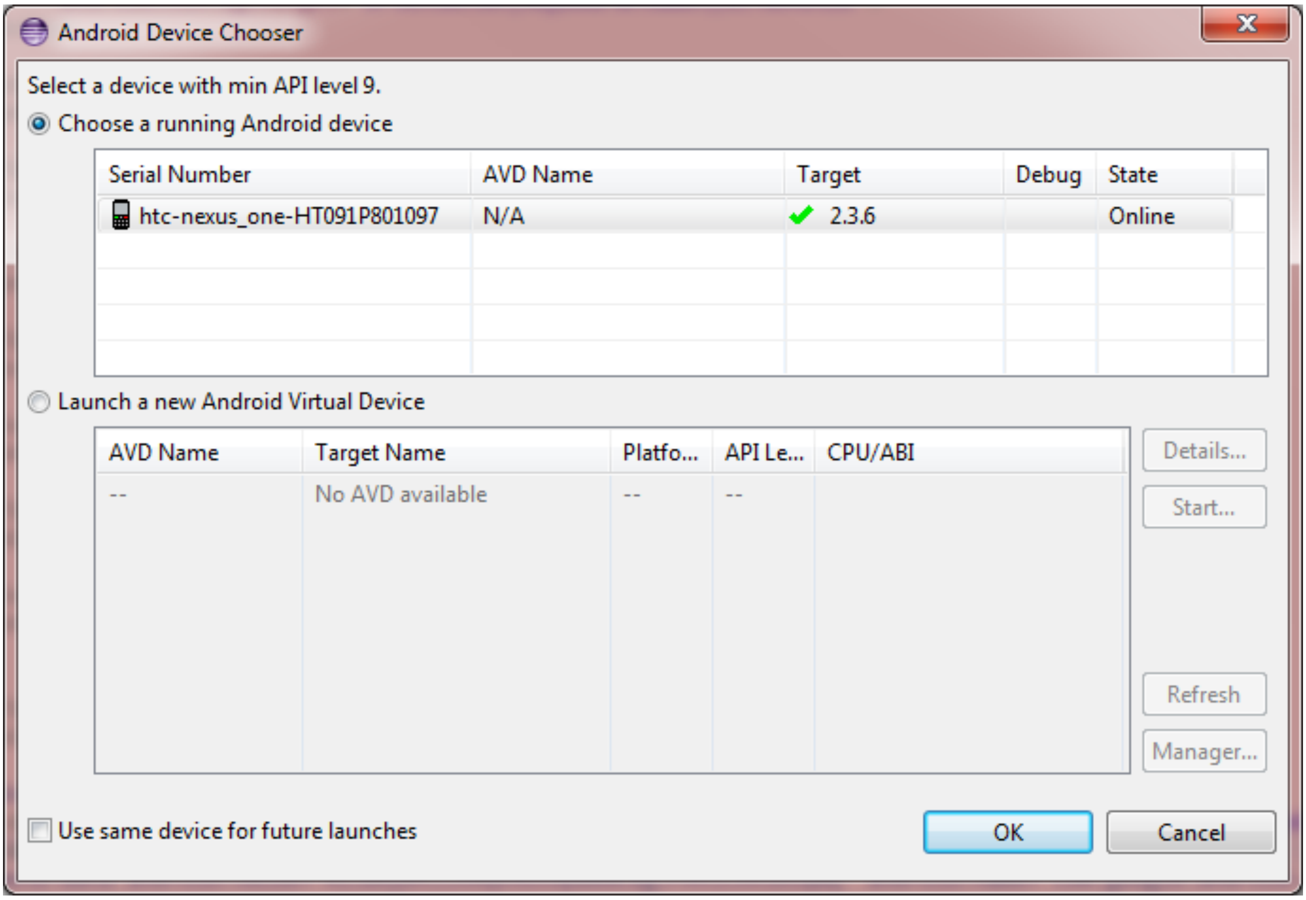
ネクサス・ワン(Nexus One)またはオプティマス・ワン(Optimus One)のような場合は、内蔵メモリーにユーザーが書き込み可能な容量がなくて外部メモリカードを追加していない場合、アプリケーションが正常にインストールされません。

Androidの仮想デバイスを用いたテストは、サポートしていません。

Android Studio



Eclipse



4.3.2 インストールファイル作成

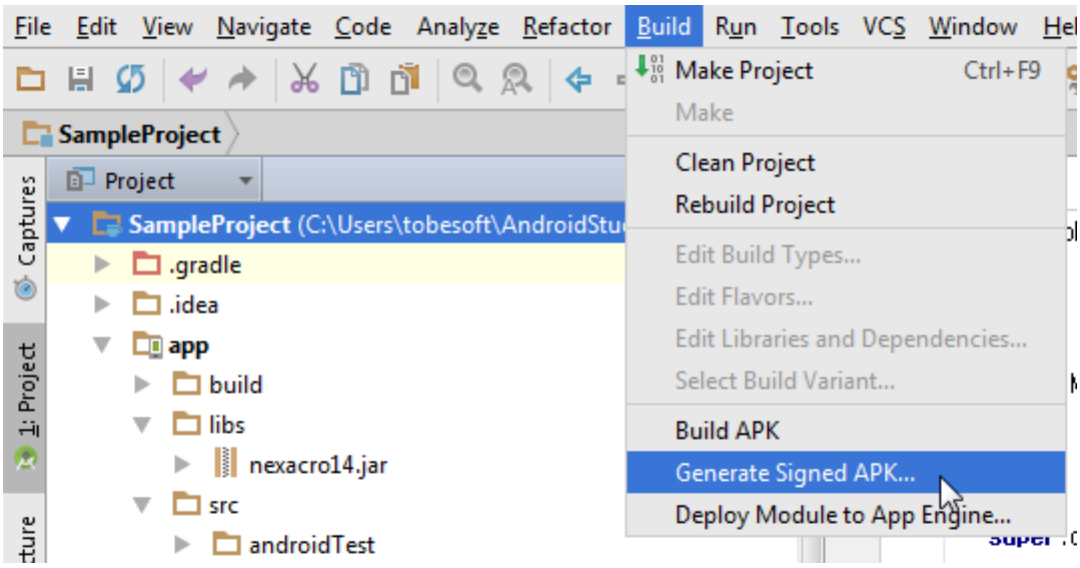
完成したアプリを配布するには、アプリを認識することができるkeystoreを作成し、署名されたAPKファイルを作成する必要があります。keystoreファイルは、最初の1回作成し、その後アプリが更新されたときは、最初に作成されたkeystoreファイルを使用します。

keystoreとSigned APKを生成する方法は、GoogleのWebページを参照してください。

> http://developer.android.com/guide/publishing/app-signing.html

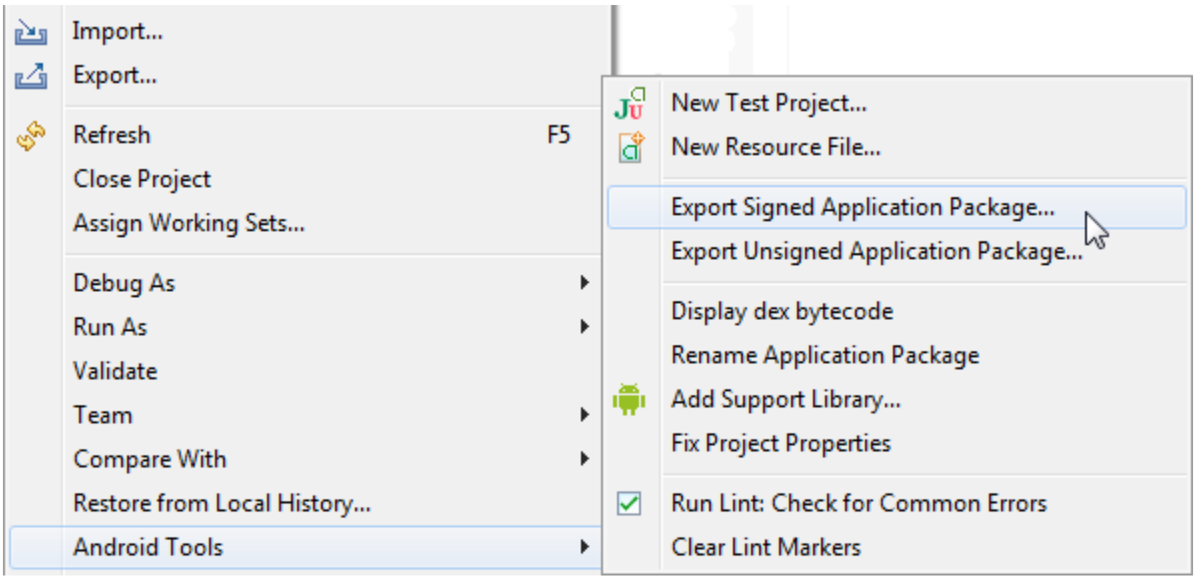
Android Studio

上部メニューから「Build > Generate Signed APK」項目を選択します。



Eclipse

プロジェクトを選択し、コンテキストメニューから[Android Tools > Export Signed Application Package]を選択します。



第5章

アプリ配布

アプリの配布は、OSとマーケット運営者に依存であるため、おおよその内容のみ記述しました。より詳細な内容は、各項目内に記述されたWebページを参照してください。

5.1　アーカイブファイル配布方法

nexacro studioで作成したアーカイブファイルを配布サーバーにコピーします。サーバーへのコピーは、nexacro studioでサポートされていないため、直接サーバーにコピーする必要があります。

コピーする場所は、各デバイスのアプリのプロジェクトで指定されたstart_android[ios].jsonパスと同じである必要があります。

次の表には、配布サーバから配布するアーカイブファイルのリストです。

配布項目	ファイル名	内容物
ArchiveInfo	start_android[ios].json	アーカイブファイル、Baseライブラリパスの情報
Base ライブラリ	nexacro14lib.zip	圧縮された形式のnexacro platformフレームワーク
プロジェクト アーカイブ ファイル	Archive00.xzip	nexacro studioから開発したnexacro platform HTML5プロジェクトファイル
テーマ アーカイブ ファイル	[テーマ名].zip	nexacro platform HTML5プロジェクトテーマ
Run	Run.zip	nexacro platformで使用するコンポーネント情報と、最初に実行するADLのJ

配布項目	ファイル名	内容物
アーカイブ ファイル		avaScriptファイルのパス情報(iOSを使用する場合にのみ適用)

5.2　Androidマーケットを通じたアプリの配布方法

5.2.1　Google Play

Google Play 配布のためには、Google Playのアカウントが必要です。

> https://play.google.com/apps/publish/signup

Google Play Webページの指示に従って生成されたAPKを登録すると配布されます。特別な検収期間はありませんし、APKに異常がなければすぐに配布することができます。

5.2.2　SamsungApps

SamsungAppsで配布するためには、SamsungAppsにsellerとして登録する必要があります。

> http://seller.samsungapps.com

sellerアカウントの作成後に、Webページの指示に従ってAPKを登録すると、検収手続きを経て配布されます。SamaungAppsは、独自の検収手続きがあり、Webページの指示に従って検査項目に問題がなければ、検収完了後、配布することができます。

5.3 プライベートサーバを介してAndroid APKを配布する方法

アンドロイドAPKは、プライベートサーバまたは電子メールを介して配布することができます。(端末からAPKをダウンロードしてインストールできるパスなら、どこにも配布可能です。)

Android端末は、デフォルト設定でマーケットを通じた正式配布ではない場合、インストールを防いでいます。したがって、プライベートサーバを利用した配布を行うには、[設定 > アプリケーション > 提供元不明のアプリ]の確認項目にチェックを入れると、Android端末上での設定が完了します。

> 端末メーカーによって、設定メニュー項目が異なることがあります。詳細については、Androidの開発者センターまたは端末のマニュアルを参照してください。
> http://developer.android.com/guide/publishing/publishing_overview.html (英語)

1. Webセーバーを介して配布

   Webサーバーにapkファイルをアップロードしておいて、該当コンテンツのAPKを参照することができるURLを取得します。取得したURLをAndroid端末ブラウザのアドレスバーに入力すると、ダウンロードしてインストールされます。

2. 電子メールを介して配布

   電子メールにAPKファイルを添付します。APKが添付された電子メールをAndroid端末で開くと、添付されたAPKをダウンロードしてインストールすることができます。

5.4　iOS APP配布方法

iOSは開発環境の設定時に作成したApple iOS開発者アカウントで配布することができます。

1. itunes connectの「Manage Your Applications」で、配布のためのAPPの情報を入力します。

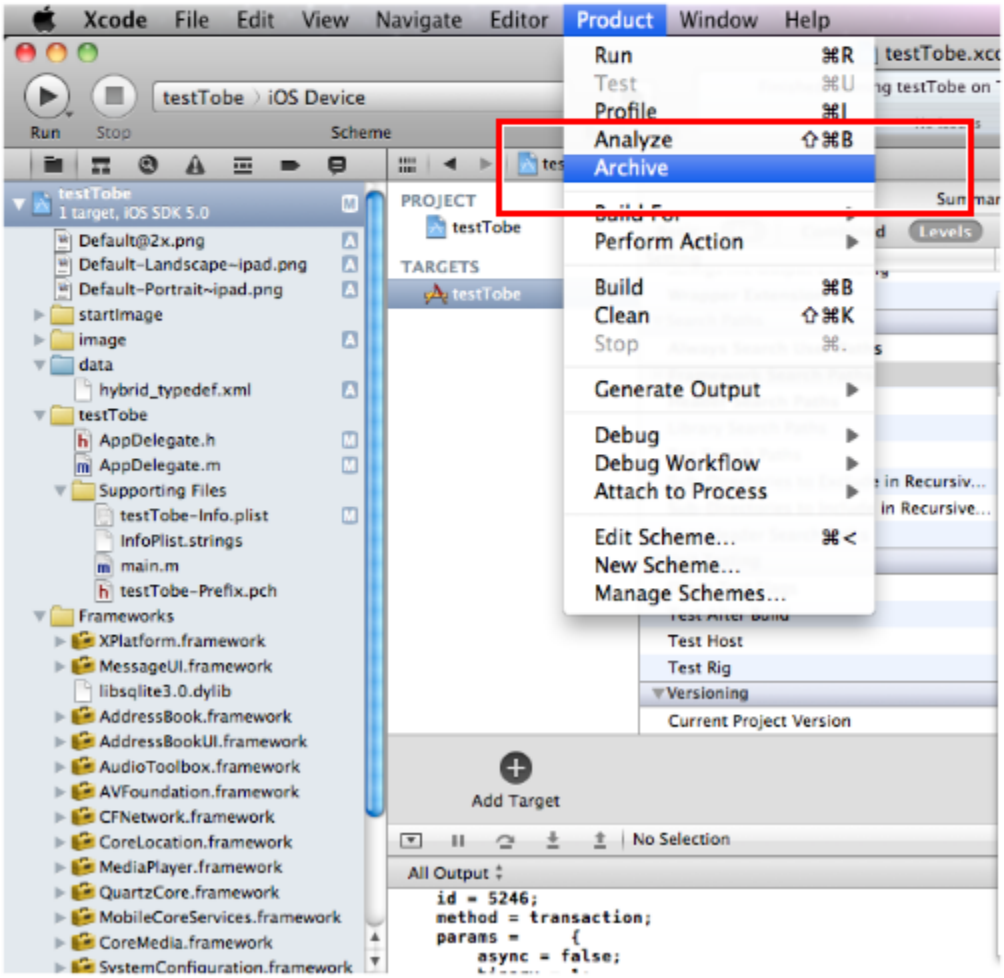
   http://itunesconnect.apple.com

2. 配布用のDistribution provisionを作成します。開発者WebページのiOS Provisioning Portalから作成することが可能であり、「how to」を参照して作成します。

   http://developer.apple.com/ios/manage/provisioningprofiles/viewDistributionProfiles.action

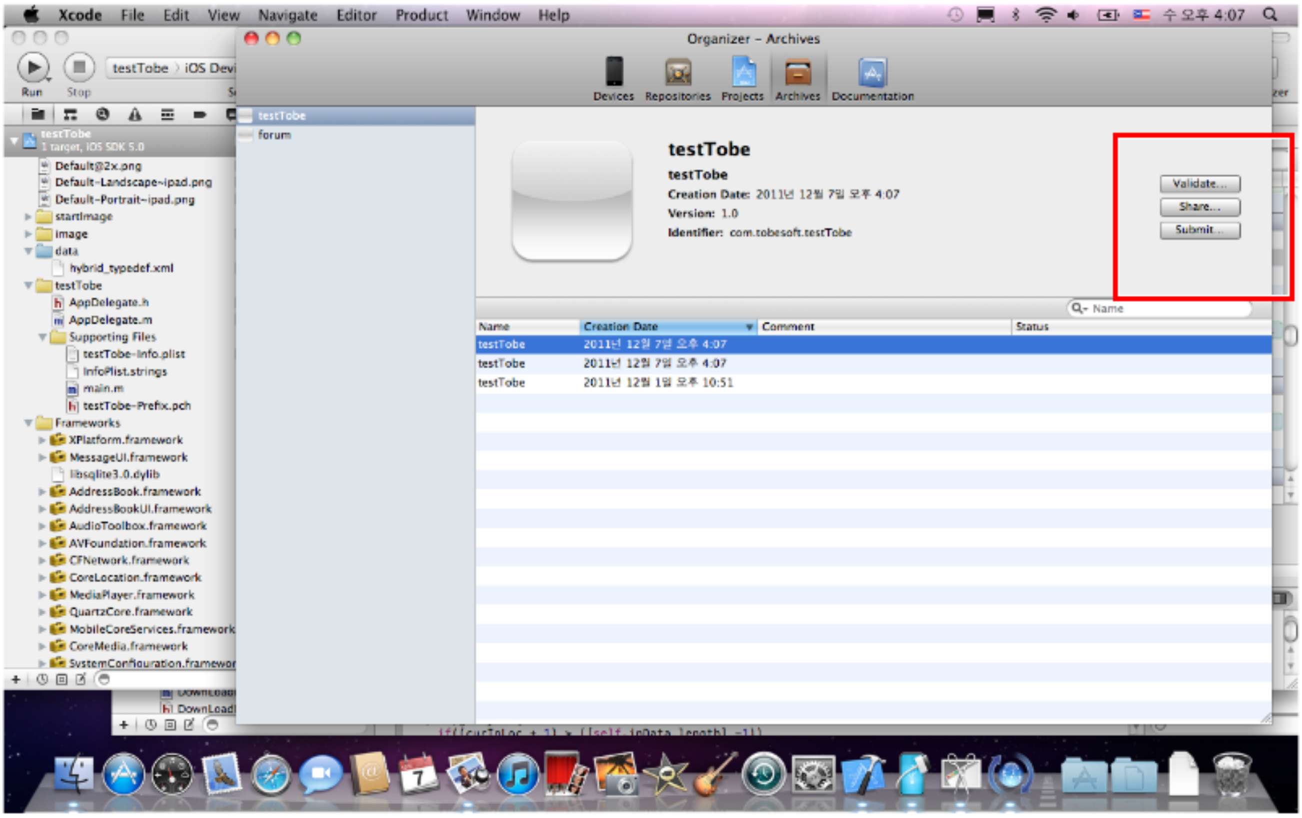
3. Xcodeの [Product > Archive] メニューで、Distribution provisionにAPPをビルドします。

   Archiveメニューを選択すると、下図のような画面が表示されます。下図でValidateは、Submitをするために必要な条件を満たしていることをご検討する際に使用し、この時に、必ずネットワークが接続されている必要があります。Shareは正式配布の前に他の端末から確認またはテストするために使用します。「Submit」をクリックして、手順に従って進みます。



> Xcodeバージョン4.3以降は、share、submitボタンの代わりにdistributeボタンのみ提供されます。distributeボタンのクリック時に進行される段階にShare機能が含まれています。

4. Xcodeの[Window > Organizer]メニューから「Archives tab」に生成されたAPPをAppstoreで配布します。
5. 配布されたAPPは、itunesconnectの「Manage Your Applications」から確認できます。
6. itunesconnectWebページを介してAppleの検査状況を確認することができます。検収完了後にAppstoreでの配布が完了します。



第IV部

参考

第1章

Generate

1.1　概要

nexacro platformで作成されたアプリケーションは、すべてHTML5コンテンツをベースに作成されます。生成されたアプリケーションをどのような形でどのような環境でユーザーが使用するかによって、実行時またはHTML5で区切られます。ランタイムは、nexacro platformランナー(Runner)でアプリケーションを実行することになり、HTML5アプリケーションは、一般的に使用するWebブラウザで該当するページのURLにアクセスする方式でアプリケーションを実行するようになります。

nexacro platformアプリケーションをnexacro studioで実行すると、コードを変換(Generate)する過程を経ることになります。このような一連の過程は、QuickViewを実行したり、アーカイブファイルを作成するときに自動的に実行され、Buildメニューから明示的に、指定のフォームや、プロジェクトの作業を実行することができます。

内部的にはnexacro studioに含まれたnexacrogenerator.exeを呼び出し、このモジュールが*.xadl, *.xfdlなどのファイルをJavaScriptファイルに変換する作業を行います。

アプリケーションの開発およびテストを完了し、配布および運用段階では、ユーザーが受信するファイルを最小化し、必要によって暗号化が必要です。このような場合nexacrocompressor.exeで提供するオプションを適用することができます。

> 生成されたスクリプトファイルは、クライアントに保存され、必要によってWebサーバー上で実行可能な場所に移動する必要があります。もし、Webサーバーがクライアントで動作している場合、nexacro studioで出力結果の位置を、ローカルのWebサーバーのパスに指定します。

対象ファイル	生成ファイル	説明
	index.html	アプリケーションを実行させるデフォルトHTMLファイル
	QuickView.html	QuickView実行時に使用するHTMLファイル

対象ファイル	生成ファイル	説明
*.xadl	*.xadl.js	ADLファイル
	*.xadl.quickview.js	QuickView実行時に使用するADLファイル
/Service/*.xfdl	/Service/*.xfdl.js	FDL(画面)ファイル
default.xtheme	_theme_/default/theme.css _theme_/default/theme.css.js _theme_/default/images	テーマとイメージファイルのフォルダ
/nexacro14lib	/nexacro14lib	オプションに応じてマージ処理

1.2 nexacrogenerator.exeの使用法

nexacrogenerator.exeは、下記のオプションを使用してコンソールから直接実行が可能です。

1.2.1 オプション

```
nexacrogenerator.exe -A -O [args...]
```

オプション	説明
-A <Target ADL Path>	GenerateするアプリケーションのTarget ADLパスを指定します。 このオプションは、必ず入力される必要があります。
-O <Output Path>	Generateされたファイルが保存されるOutputパスを指定します。 指定されたパスが存在しない場合、自動的にパスを作成します。 このオプションは、必ず入力される必要があります。
-P <Project Name>	Generateされたファイルを保存するプロジェクト名を指定します。 入力されたプロジェクト名は-Oの入力値に追加され、ファイルが保存されるパスとして指定されます。 入力しない場合は、-Aに入力されたXADLファイル名がプロジェクト名に自動的に指定されます。
-B <library Path>	Generateに必要なライブラリファイルがあるパスを指定します。 入力しないと、"nexacro.exe"がインストールされたパスにあるデフォルトライブラリファイルを使用します。
-M	Mergeオプションを適用してジェネレートされます。
-AO	Moduleファイルのみジェネレートするときに使用します。 -Mオプションに応じて、マージを実行します。 注意：-AO、-S、-F、-Tと一緒に使用することができません。
-MO	Moduleファイルのみジェネレートするときに使用します。 -Mオプションに応じて、マージを実行します。
-S <Service ID>	GenerateするService Prefix IDを指定します。 TypeDefinitionで定義したService Prefix IDのみを指定することができます。 注意：-AO、-F、-Tと一緒に使用することができません。
-F <path1>,[<path2>,…]	Generateするファイルのパスのリストを指定します。 複数の入力時 ','(Comma)または ';'(Semicolon)で区切ります。 注意：-AO、-S、-Tと一緒に使用することができません。

オプション	説明
-T <themeid1>,[<themeid2>,…]	GenerateするTheme IDを指定します。 複数の入力時 ','(Comma)または ';'(Semicolon)で区切ります。 注意：-AO、-S、-Fと一緒に使用することができません。
-TABORDER	Generate時にコンポーネントをTab Order順に生成します。 入力しないとZ-Order順に生成します。
-L <Path>	Generateの進行過程を保存するログファイルのパスを指定します。 指定されたパスが存在しない場合、自動的にパスを作成します。 ログファイルの作成時にGenerate実行時間が増加することがあります。
-H	"nexacrogenerator.exe"コマンドのヘルプ情報を提供します。

-Hオプションは、ユーザーのシステム設定によって、英語、日本語、韓国語をサポートします。

1.2.2　オプション使用例

Application全体をGenerateする場合

「C:\TestGenerate\TestGenerate.xadl」ファイルを基準にアプリケーションをGenerateて「E:\ResultGenerate」に出力します。
アプリケーションの実行に必要な実行ファイルとテーマファイル、モジュールファイルも一緒に生成されます。

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O "E:\
ResultGenerate"
```

```
E:\ResultGenerate\TestGenerate\index.html
E:\ResultGenerate\TestGenerate\QuickView.html
E:\ResultGenerate\TestGenerate\popup.html
E:\ResultGenerate\TestGenerate\start.json
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.js
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.quickview.js
E:\ResultGenerate\TestGenerate\_theme\…
E:\ResultGenerate\TestGenerate\Base\…
…
E:\ResultGenerate\TestGenerate\nexacro14lib\framework\Platform.js
E:\ResultGenerate\TestGenerate\nexacro14lib\framework\Systembase.js
…
E:\ResultGenerate\TestGenerate\nexacro14lib\component\ComComp\Button.js
E:\ResultGenerate\TestGenerate\nexacro14lib\component\ComComp\Combo.js
…
```

-Mオプションを追加した場合には、モジュールファイルをコンポーネント、オブジェクトごとに生成せずに、各グループごとに圧縮して生成します。

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O "E:\
ResultGenerate" -M
```

```
E:\ResultGenerate\TestGenerate\index.html
E:\ResultGenerate\TestGenerate\QuickView.html
E:\ResultGenerate\TestGenerate\popup.html
E:\ResultGenerate\TestGenerate\start.json
```

```
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.js
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.quickview.js
E:\ResultGenerate\TestGenerate\_theme\…
E:\ResultGenerate\TestGenerate\Base\…
…
E:\ResultGenerate\TestGenerate\nexacro14lib\framework\Framework.js
E:\ResultGenerate\TestGenerate\nexacro14lib\component\ComCom.js
E:\ResultGenerate\TestGenerate\nexacro14lib\component\CompBase.js
…
```

Generate時OutputパスにXADL名ではない使用者が指定した名前にパスを指定する場合

"C:\TestGenerate\TestGenerate.xadl"ファイルを基準にGenerateする時、Projectパス名を"TestGenerate2"に変更して"E:\ResultGenerate"に出力します。
アプリケーションの実行に必要な実行ファイルとThemeファイルも一緒に生成されます。

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O "E:\ResultGenerate" -P "TestGenerate2"
```

```
E:\ResultGenerate\TestGenerate2\index.html
E:\ResultGenerate\TestGenerate2\QuickView.html
E:\ResultGenerate\TestGenerate2\popup.html
E:\ResultGenerate\TestGenerate2\start.json
E:\ResultGenerate\TestGenerate2\TestGenerate.xadl.js
E:\ResultGenerate\TestGenerate2\TestGenerate.xadl.quickview.js
E:\ResultGenerate\TestGenerate2\_theme\…
E:\ResultGenerate\TestGenerate2\Base\…
…
```

Generate時に別のライブラリパスを指定する場合

"C:\TestGenerate\TestGenerate.xadl"ファイルを基準にアプリケーションをGenerateする時、参照するライブラリパスを "C:\nexacro14lib"で指定し、"E:\ResultGenerate"に出力します。
ライブラリパスを指定しなかった場合には、 "nexacro.exe" インストールパスのライブラリを参照してGenerateします。

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O "E:\
ResultGenerate" -B "C:\nexacro14lib"
```

```
E:\ResultGenerate\TestGenerate\index.html
E:\ResultGenerate\TestGenerate\QuickView.html
E:\ResultGenerate\TestGenerate\popup.html
E:\ResultGenerate\TestGenerate\start.json
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.js
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.quickview.js
E:\ResultGenerate\TestGenerate\_theme\…
E:\ResultGenerate\TestGenerate\Base\…
…
```

特定XADLファイルのみ変換する場合

"C:\TestGenerate\TestGenerate.xadl"に入力されたXADLファイルのみGenerateして、 "E:\ResultGenerate"に出力します。
アプリケーションの実行に必要な実行ファイルも一緒に生成され、Themeファイルは生成されません。
そして、-T、-S、-F、-AOコマンドは、同時に使用することができません。

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O "E:\
ResultGenerate" -AO
```

```
E:\ResultGenerate\TestGenerate\index.html
E:\ResultGenerate\TestGenerate\QuickView.html
E:\ResultGenerate\TestGenerate\popup.html
E:\ResultGenerate\TestGenerate\start.json
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.js
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.quickview.js
```

特定サービスパスのみを変換する場合

"C:\TestGenerate\TestGenerate.xadl"ファイルを基準にTypeDefinitionファイルに定義されたService Prefix IDでGenerateして、"E:\ResultGenerate"に出力します。
Service IDは、単一の入力のみ可能です。
ただし、-Sの入力値は必ずTypeDefinitionに定義されたService Prefix IDである必要があります。
そして、-T、-S、-F、-AOコマンドは、同時に使用することができません。

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O "E:\
ResultGenerate" -S "Base"
```

```
E:\ResultGenerate\TestGenerate\Base\…
E:\ResultGenerate\TestGenerate\Base\…
…
```

特定ファイルを変換する場合

"C:\TestGenerate\TestGenerate.xadl"ファイルを基準に"C:\TestGenerate\Base\form1.xfdl", "C:\TestGenerate\Base\form2.xfdl"ファイルをGenerateして"E:\ResultGenerate"に出力します。
ファイルは単一、または複数の入力が可能です。
ただし、-Fの入力値にXADL、XTHEMEファイルは使用できません。
そして、-T、-S、-F、-AOコマンドは、同時に使用することができません。

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O "E:\
ResultGenerate" -F "C:\TestGenerate\Base\form1.xfdl", "C:\TestGenerate\Base\
form2.xfdl"
```

```
E:\ResultGenerate\TestGenerate\Base\form1.xfdl.js
E:\ResultGenerate\TestGenerate\Base\form2.xfdl.js
```

特定XTHEMEのみGenerateする場合

case 1) Cacheフォルダに登録されたTheme

"C:\TestGenerate\TestGenerate.xadl"ファイルを基準に"%USERAPP%\Cache\Theme\Gray.xtheme"ファイルをGenerateして"E:\ResultGenerate"に出力します。
Cacheフォルダに登録されたThemeをGenerateする場合には"Gray.xtheme"で入力します。(相対パスを指定せずにTheme名のみ入力する場合にはCacheフォルダに登録されたThemeとして認識します。)

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O "E:\
ResultGenerate" -T "Gray.xtheme"
```

```
E:\ResultGenerate\TestGenerate\_theme_\Gray\theme.css.js
E:\ResultGenerate\TestGenerate\_theme_\Gray\images\...
…
```

case 2) XADL基準に相対パスまたはPrefixパスに登録されたTheme

"C:\TestGenerate\TestGenerate.xadl"ファイルを基準に"C:\TestGenerate\Themes\NewTheme1.xtheme"ファイルをGenerateして、"E:\ResultGenerate"に出力します。
使用者が指定したパスに存在するThemeをGenerateする場合には、XADL基準に相対パスまたはPrefixパスで入力します。

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O "E:\
ResultGenerate" -T "Themes\NewTheme1.xtheme"
```

```
E:\ResultGenerate\TestGenerate\_theme_\Themes\NewTheme1\theme.css.js
E:\ResultGenerate\TestGenerate\_theme_\Themes\NewTheme1\images\...
…
```

case 3) case 1とcase 2の混合

"C:\TestGenerate\TestGenerate.xadl"ファイルを基準に"%USERAPP%\Cache\Theme\Gray.xtheme"、"C:\TestGenerate\Themes\NewTheme1.xtheme"ファイルをGenerateして、"E:\ResultGenerate"に出力します。

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O "E:\
ResultGenerate" -T "Gray.xtheme,Themes\NewTheme1.xtheme"
```

```
E:\ResultGenerate\TestGenerate\_theme_\Gray\theme.css.js
E:\ResultGenerate\TestGenerate\_theme_\Gray\images\...
…
E:\ResultGenerate\TestGenerate\_theme_\Themes\NewTheme1\theme.css.js
E:\ResultGenerate\TestGenerate\_theme_\Themes\NewTheme1\images\...
…
```

モジュールファイルのみを変換する場合

case 1) Mergeオプションを適用しない。

「C:\TestGenerate\TestGenerate.xadl」ファイルに基づいて、下位のnexacro14libの中のモジュールファイルをコピーします。

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O "E:\
ResultGenerate" -MO
```

```
E:\ResultGenerate\TestGenerate\nexacro14lib\framework\Platform.js
E:\ResultGenerate\TestGenerate\nexacro14lib\framework\Systembase.js
…
E:\ResultGenerate\TestGenerate\nexacro14lib\component\ComComp\Button.js
E:\ResultGenerate\TestGenerate\nexacro14lib\component\ComComp\Combo.js
…
```

case 2) 特定のパスのモジュールファイルを使用する。

-Bオプションで指定されたパスに基づいて、モジュールファイルをコピーします。

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O "E:\
ResultGenerate" -MO  -B "C:\nexacro14lib"
```

```
E:\ResultGenerate\TestGenerate\nexacro14lib\framework\Platform.js
E:\ResultGenerate\TestGenerate\nexacro14lib\framework\Systembase.js
…
E:\ResultGenerate\TestGenerate\nexacro14lib\component\ComComp\Button.js
E:\ResultGenerate\TestGenerate\nexacro14lib\component\ComComp\Combo.js
…
```

case 3) Mergeオプションを適用する。

「C:\TestGenerate\TestGenerate.xadl」ファイルに基づいて、下位のnexacro14libのモジュールファイルを圧縮されたファイルに変換します。

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O "E:\
ResultGenerate" -MO -M
```

```
E:\ResultGenerate\TestGenerate\nexacro14lib\framework\Framework.js
E:\ResultGenerate\TestGenerate\nexacro14lib\component\ComCom.js
E:\ResultGenerate\TestGenerate\nexacro14lib\component\CompBase.js
…
```

変換時のコンポーネントの作成順序をtaborder値に基づいてソートした後に変換する場合

「C:\TestGenerate\TestGenerate.xadl」ファイルに基づいて、アプリケーションを変換して「E:\ResultGenerate」に出力します。
xfdlファイルを変換するとき、コンポーネントのtaborder値の順にソートして変換します。
-Sまたは-Fコマンドを使用した場合にも-TABORDERコマンドを使用することができます。

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O "E:\
ResultGenerate" -TABORDER
```

```
E:\ResultGenerate\TestGenerate\index.html
E:\ResultGenerate\TestGenerate\QuickView.html
E:\ResultGenerate\TestGenerate\popup.html
E:\ResultGenerate\TestGenerate\start.json
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.js
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.quickview.js
E:\ResultGenerate\TestGenerate\_theme\…
E:\ResultGenerate\TestGenerate\Base\…
…
```

Generate結果をLogファイルに保存する場合

"C:\TestGenerate\TestGenerate.xadl"ファイルを基準にアプリケーションをGenerateして、"E:\ResultGenerate"に出力し、ログの内容を"E:\ResultGenerate\Log\TestGenerate_1.1.01.txt"に保存します。

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O "E:\
ResultGenerate" -L "E:\ResultGenerate\Log\TestGenerate_1.1.01.txt"
```

```
E:\ResultGenerate\TestGenerate\index.html
E:\ResultGenerate\TestGenerate\QuickView.html
E:\ResultGenerate\TestGenerate\popup.html
E:\ResultGenerate\TestGenerate\start.json
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.js
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.quickview.js
E:\ResultGenerate\TestGenerate\_theme\…
E:\ResultGenerate\TestGenerate\Base\…
…
E:\ResultGenerate\Log\TestGenerate_1.1.01.txt
```

1.3 nexacrocompressor.exe

nexacrocompressor.exeは、以下の表のオプションを使用して、コンソールから直接実行することもできます。

1.3.1 オプション

```
nexacrocompressor.exe -F -O [args...]
nexacrocompressor.exe -P -O [args...]
```

オプション	説明
'-F <file1>,[<file2>,…]	Compressするファイルパスを指定します。 複数で入力時 ','(Comma)または ';'(Semicolon)で区切ります。 -Fまたは-Pオプションは、必ず入力する必要があり、重複して使用することができません。 注意：-Fと-Oは、ファイルの順序と数が一致する必要があります。
-P <directory>	Compressするディレクトリのパスを指定します。 -Fまたは-Pオプションは、必ず入力する必要があり、重複して使用することができません。 注意：-Pと-Oには、それぞれ１つのディレクトリのみを指定する必要があります。
-O <file1>,[<file2>,…]	-Fでファイルを指定した場合 Compressされたファイルが保存されるOutputパスを指定します。 複数で入力時 ','(Comma)または ';'(Semicolon)で区切ります。 注意：-Fを使用する場合、-Fと-Oはファイルの順序と数が一致する必要があります。
-O <directory>	-Pでディレクトリを指定した場合 Compressされたファイルが保存されるOutputパスを指定します。 注意：-Pを使用する場合、-Pと-Oはファイルの順序と数が一致する必要があります。 指定されたパスが存在しない場合、自動的にパスを作成します。 このオプションは、必ず入力される必要があります。 注意：-Fを使用する場合、-Fと-Oはファイルの数が一致する必要があります。
-S <string>	Compressされたファイル名にサフィックスを指定します。
-Shrink	変数名を難読化します。 eval構文を含む関数は難読化しません。

オプション	説明
-IgnoreEval	eval関数に関係なく、難読化します。（推奨しない） 注意：このオプションを使用するには、-Shrinkオプションを先に使用する必要があります。
-Compile	nexacro Runtimeで使用されている暗号化されたファイルに変換します。 ※ このオプションを使用して取得した結果物は、nexacro Runtimeでのみ使用可能です。 該当オプションの後にパスを指定してコンパイルされたファイルを別途に保存することができます。 該当オプションの後にパスを指定しない場合、-Oオプションに該当するパスにコンパイルされたファイルが生成されます。
-Compile [<file1>,<file2>,...]	-Fオプションでファイルを指定した場合 暗号化されたファイルが保存される Outputパスを指定します。 複数のファイルを入力するときは「、」（カンマ）または「;」(セミコロン)で区切ります。 ※ -Fオプションを使用する場合に -Fと -Compileはファイルの順序と数が一致する必要があります。 ※ -Fオプションを使用する場合、XMLファイル（拡張子が「.xml」であるファイル）もコンパイル対象に含みます。
-Compile [<directory>]	-Pオプションでディレクトリを指定した場合 暗号化されたファイルが保存される Outputパスを指定します。 ※ -Pオプションを使用する場合、-Pと -Compileはそれぞれ一つのディレクトリだけを指定する必要があります。
-Ignorecompress <.ignorecompress>	Compressせずに処理する必要があるリストを保存した.ignorecompressファイルのパスを指定します。 注意：-Pを使用する場合にのみ適用されるオプションです。
-H	"nexacrocompressor.exe"コマンドのヘルプ情報を提供します。

1.3.2　オプション使用例

特定のjsファイルをCompressする場合

"C:\JS\test1.js, "C:\JS\childdir\test3.js"ファイルをCompressして"E:\JS\test1.js", "E:\JS\childdir\test3.js"で出力します。
コメント/空白を削除

```
nexacrocompressor.exe -F "C:\JS\test1.js,C:\JS\childdir\test3.js" -O "E:\JS\test1.js,E:\JS\childdir\test3.js"
```

```
E:\JS\test1.js
E:\JS\childdir\test3.js
```

特定のディレクトリに存在するjsファイルをCompressする場合

"C:\JS"ディレクトリに存在するjsファイルをCompressして"E:\JS"ディレクトリに出力します。
コメント/空白を削除

```
nexacrocompressor.exe -P "C:\JS" -O "E:\JS"
```

```
E:\JS\test.js
E:\JS\test1.js
E:\JS\test2.js
E:\JS\childdir\test3.js
E:\JS\childdir\test4.js
```

Compress時、Outputファイル名に接尾辞を指定する場合

"C:\JS\test1.js"ファイルをCompressして"E:\JS\test1.js"で出力するとき、接尾辞"_2014100601"を追加します。
コメント/空白を削除

```
nexacrocompressor.exe -F "C:\JS\test1.js" -O "E:\JS\test1.js" -S "_2014100601"
```

```
E:\JS\test1_2014100601.js
```

Compress時Outputファイルに変数難読化を適用する場合

"C:\JS\test1.js"ファイルをCompressして"E:\JS\test1.js"で出力するとき、接尾辞"_2014100601"を追加します。

コメント/空白を削除
ただし、関数の内部でeval()関数を使用した場合、該当関数は、難読化対象から除外します。

```
nexacrocompressor.exe -F "C:\JS\test1.js" -O "E:\JS\test1.js" -Shrink
```

```
E:\JS\test1.js
```

圧縮時Outputファイルに変数難読化を適用する際、難読化の範囲でeval関数を無視する場合

"C:\JS\test1.js"ファイルをCompressする時、変数難読化も適用しますがeval関数の内部はスキップして"E:\JS\test1.js"に出力します。

コメント/空白を削除
eval()関数と関係なく、すべて実行します。

```
nexacrocompressor.exe -F "C:\JS\test1.js" -O "E:\JS\test1.js" -Shrink -IgnoreEval
```

```
E:\JS\test1.js
```

圧縮時対象リストから特定のファイルを除外する場合

ディレクトリに存在するjsファイルを圧縮して、"E:\JS"ディレクトリに出力します。この過程で-IgnoreCompressオプションの値として入力した.IgnoreCompressファイルに定義されたファイルは、圧縮対象から除外します。
コメントと空白は削除します。（.IgnoreCompressファイルに定義されたファイルは除外します。）

```
nexacrocompressor.exe -P "C:\JS" -O "E:\JS" -IgnoreCompress "C:\nexacro14\Project\.IgnoreCompress"
```

```
<?xml version="1.0" encoding="utf-8"?>
<CompressConfigue version="1.0">
    <IgnoreCompress>
        <File path=".\test1.js"/>
        <File path=".\Childdir\test4.js"/>
    </IgnoreCompress>
</CompressConfigue>
```

```
E:\JS\test.js
E:\JS\test1.js (非圧縮)
E:\JS\test2.js
E:\JS\childdir\test3.js
E:\JS\childdir\test4.js (非圧縮)
```

第2章

応用プログラムのキャッシュ

開発者が開発した応用プログラムは、配布サーバーから毎回ダウンロードするのではなく、ネットワークを効率的に使用するためにキャッシュを使用します。

2.1　キャッシュの種類

nexacro platformがサポートするキャッシュの種類は以下の通りです。

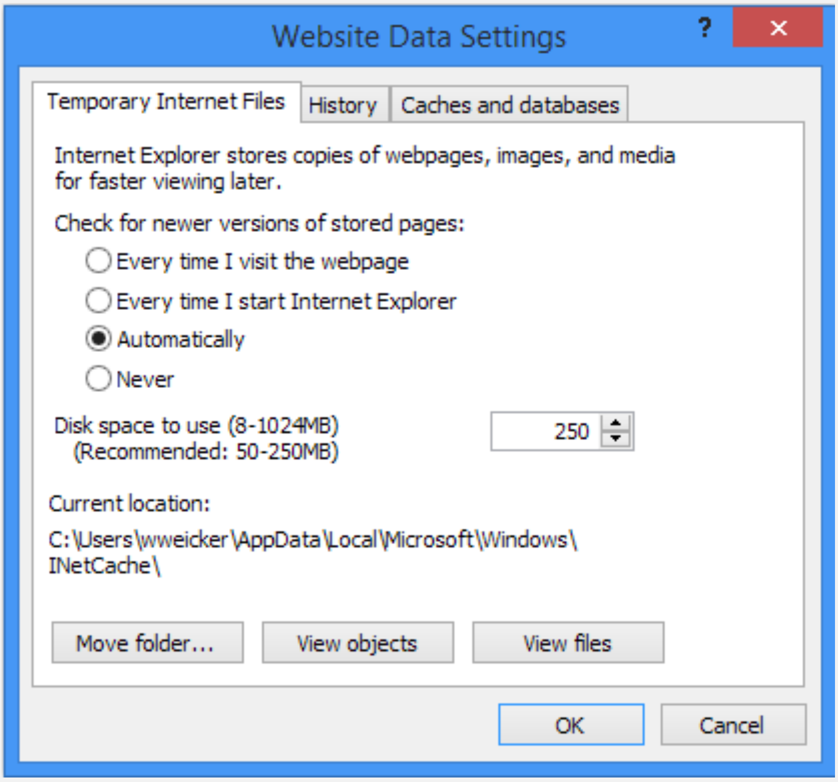
キャッシュの種類	説明
none	キャッシュ機能を使用しません。
dynamic	サーバーから受信したファイルが更新された時にのみ受信します。 サーバーのファイルが更新されない場合には、Local Cacheファイルを再利用します。
session	nexacro platformエンジンを起動する時に1度だけ受信し(もしもLocal Cacheファイルと一致すれば再利用)、該当のエンジンが終了するまでLocal Cacheファイルのみを使用します。
static	サーバーから1度でも受信したこのとあるファイル対し、nexacro platformエンジンの再実行と関係なくLocal Cacheファイルのみを使用します。 このため、再受信したい場合にはバージョンアップしてください。

cachelevelを指定しなかった場合、デフォルト Cacheプロパティは Dynamicに適用されます。

プロパティ値が「static」の場合、キャッシュファイルをローカルストレージに保存することになりますが、ウェブブラウザの機能制限によって、ローカルストレージが使用できません。それで、HTML5バージョンを使用している場合には、プロパティ値が「static」の場合にも、「session」で動作します。

HTML5バージョンを使用する場合には、プロパティ値が「none」であれば、ブラウザのキャッシュを使用しないのが正常です。Firefoxを使用する場合には、ブラウザのキャッシュを使用する症状がありますが、この症状はFirefoxのバグです。
https://bugzilla.mozilla.org/show_bug.cgi?id=1129500

HTML5バージョンのアプリケーションがMS IEから実行される場合、nexacro platformの内部で利用するキャッシュメモリよりも、インターネットエクスプローラのキャッシュが優先動作する場合があります。 ウェブブラウザのキャッシュオプションが「自動的に確認する(A)」であるときに、サーバーから変更されたJavaScriptファイル（*.js）が更新されない現象は、制約事項です。



ブラウザを新たに起動したり refreshする動作では、製品が管理するメモリ上の cacheがすべてクリアされるため、session / dynamic / static設定の場合、設定値とは無関係にブラウザの動作に依存するようになります。

nexacro studioで「Launch Project」または「Quick View」メニューを使用してRuntimeを実行する場合には、typedefinitionに設定されているService-cachelevel値を無視してnoneに動作します。

(cachelevelプロパティの設定値が変更されることではありません。)

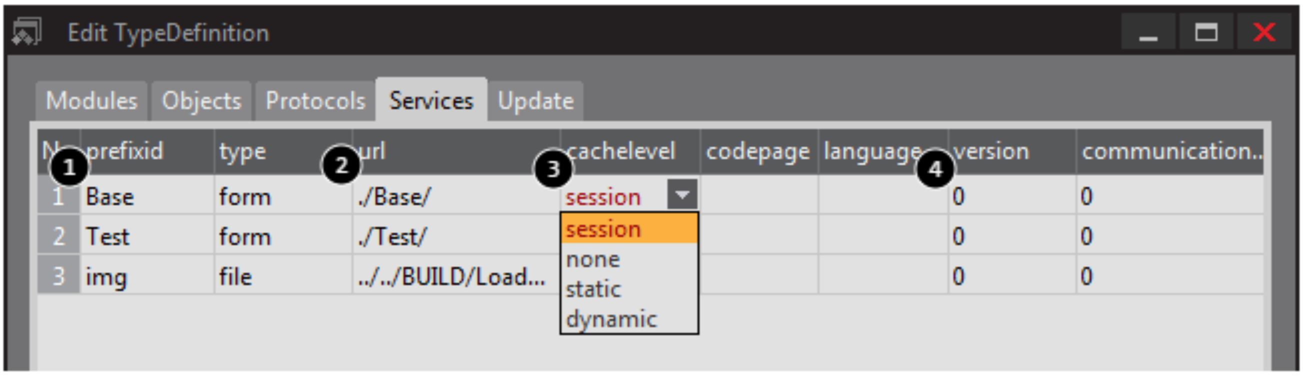
nexacro platformで画像を処理するときには、画像表現に関連する機能(size、stretchなど)とパフォーマンス上の理由で使用ブラウザのキャッシュ設定に従う動作をするように設計されました。
(プロジェクトserviceグループに設定されたcachelevelとは別の動作です。)

ExcelImportObjectまたはExcelExportObjectを使用してExcelファイルをインポートまたはエクスポートする場合には、指定されたcachelevelプロパティ値に関係なく、無条件noneに動作します。

2.2 キャッシュの適用方法

2.2.1 cachelevel

以下は、nexacro studioのType Definition Editor画面です。



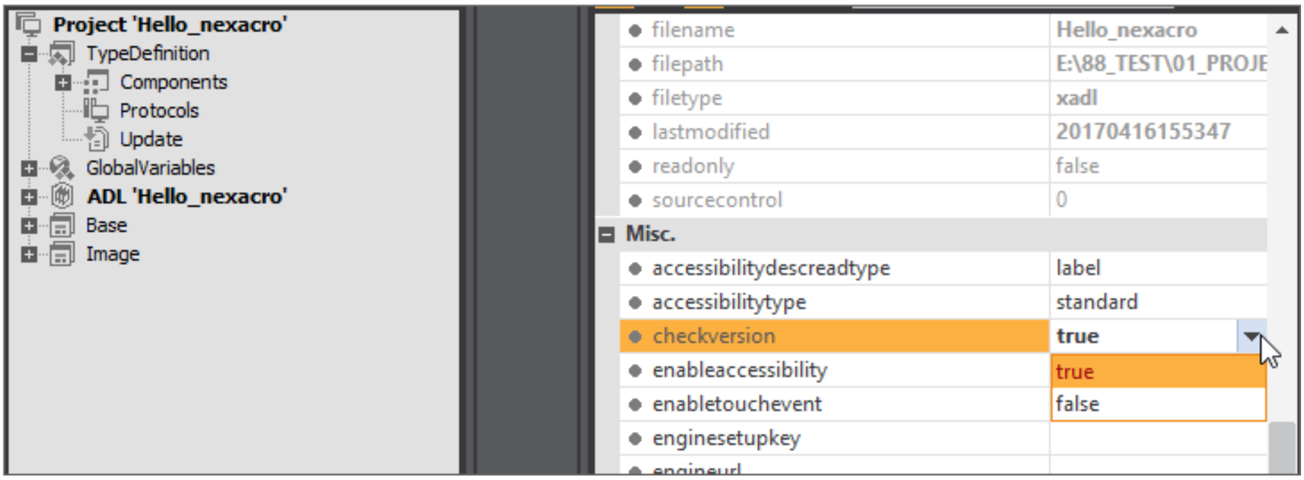
1. Cacheはここで登録したサービスグループにのみ適用されます。
2. ここでは、“./Base/”というディレクトリにある全てのファイルに対し“none” cacheが適用されることがわかります。
3. キャッシュの種類を選択します。

4. この他にもこの画面でバージョンを設定し、staticである場合も追加受信ができるようにすることができます。

> Script
> ```
> application.services["Base"].set_cachelevel("none" )
> ```

2.2.2 checkversion

ADLのcheckversionプロパティ値を「true」に設定すると、製品のバージョンや指定されたversion値を変更した際に、強制的にキャッシュを適用しないようにすることができます。



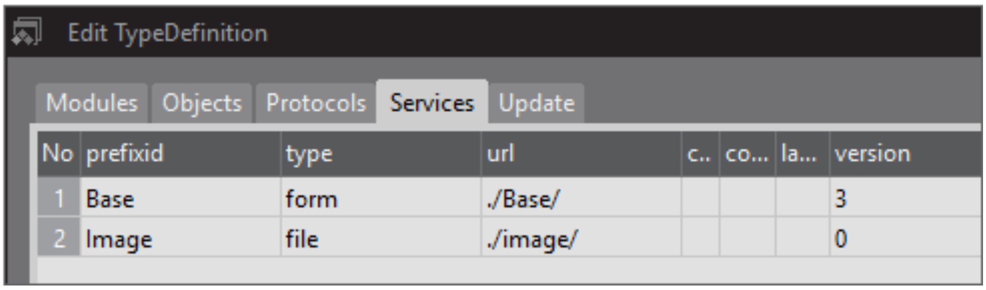
checkversionプロパティ値を「true」に設定した場合には、generateを行う際に生成されるコードの一部が以下のように変更されます。

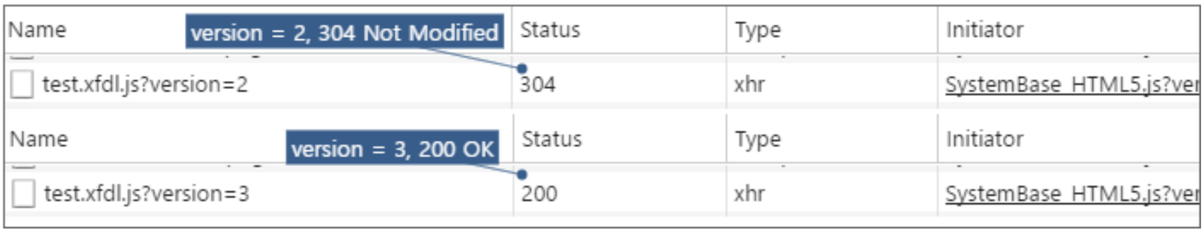
- QuickView.html, index.html, popup.html
    - nexacro platformのライブラリ・スクリプトとADLファイルを呼び出します。その際に、URLパラメータ（Query String）が追加された状態で呼び出します。
    - nexacro platformのライブラリ・スクリプトでは、製品のバージョンをURLパラメータとして使用します。
    - ADLファイルでは、ADLのversionプロパティ値をURLパラメータとして使用します。

> 製品のバージョンは、別途入力しません。
> インストール済のnexacro studioのバージョンに基づいて設定されます。

```
<meta http-equiv="pragma" content="no-cache">
      <script type="text/javascript"
        src="./nexacro14lib/framework/SystemBase.js?version=14.0.1.1210"
></script>
...
  <script type="text/javascript">
    function oninitframework()
    {
        application.load("Hello_nexacro",
            "Hello_nexacro.xadl.quickview.js?version=1.0.1");
```

- Formファイル呼び出し
    - ADLファイルのロード後、Formファイルの呼び出し時に、TypeDefinitionのサービスグループで指定したversionの設定値をURLパラメータとして使用します。
    - サービスグループからversionを新たに設定した場合には、cachelevelの設定値は無効となり、「none」として動作します。
    - ネットワーク通信状況を確認すると、versionの設定値を変更した際にキャッシュされないことが確認できます。





第3章

Dataset XML Format

nexacro platformはサーバーモジュールとしてX-APIを提供します。このAPIを使って生成された Dataset format を nexacro platform HTML5が解釈してUI画面に表示します。

ここでは X-APIが生成するDataset formatを説明します。このformatを遵守したXML Formatファイルはnexacro platform HTML5が認識できるデータです。すなわちX-APIを使わなくてもサーバープログラムを開発することはできます。

3.1 Dataset XML layout

XML Declaration					
Root					
	Cache				
	Parameters				
		Parameter (繰り返し)			
	Dataset (繰り返し)				
		ColumnInfo			
			ConstColumn (繰り返し)		
			Column (繰り返し)		
		Rows			
			Row (繰り返し)		
				Col (繰り返し)	
				OrgRow	
					Col (繰り返し)

3.1.1 XML 宣言

xmlであることを明示するxml 宣言文を、次のように定義します。

```
<?xml version="1.0" encoding="utf-8"?>
```

xml 宣言文はxml文書の一番前になければならず、この宣言文の前には空白文字を含めて、いかなる文字もあってはなりません。

3.1.2 xml 例

```
<?xml version="1.0" encoding="utf-8"?>
<Root xmlns="http://www.nexacroplatform.com/platform/dataset" ver="4000" >
     <Parameters>
          <Parameter id="service">stock</Parameter>
          <Parameter id="method">search</Parameter>
     </Parameters>
     <Dataset id="output">
          <ColumnInfo>
               <ConstColumn id="market" size="10" type="STRING" value="kse"/>
               <ConstColumn id="openprice" size="10" type="INT"
               value="15000"/>
               <Column id="stockCode" size="5" type="STRING"/>
               <Column id="currentprice" size="10" type="INT"/>
          </ColumnInfo>
          <Rows>
               <Row>
                    <Col id="currentCode">10001</Col>
                    <Col id="currentprice">5700</Col>
               </Row>
               <Row>
                    <Col id="currentCode">10002</Col>
                    <Col id="currentprice">14500</Col>
               </Row>
          </Rows>
     </Dataset>
```

```
</Root>
```

3.2 Dataset エレメント 説明

Datasetのエレメント(element)を説明します。

3.2.1 Root

概要

Datasetを現わす最上位エレメント

子ノード

Parameters, Dataset

繰り返し指定

必ず、1つは存在する必要があります。

属性

```
<Root xmlns="http://www.nexacroplatform.com/platform/dataset">
<Root xmlns:nexacro="http://www.nexacroplatform.com/platform/dataset">
<Root xmlns=".." version="1000">
```

属性名	説明
xmlns	ネームスペース、もし基本ネームスペースを使わない場合なら、 xmlns:nexacroと明示します。
ver	Dataset Layoutのバージョンを明示します。

3.2.2 Parameters

概要

通信時に必要なパラメータを明示するためのエレメント。実際にパラメータ値を有するのは子ノードParameterであり、Parameters エレメントとは、子ノードParameterを有する集合の役割です。

子ノード

Parameter

繰り返し指定

存在しないか、1つ存在する場合があります。

3.2.3 Parameters > Parameter

概要

パラメータの値を明示します。

子ノード

なし

繰り返し指定

存在しないか、パラメータエレメントはパラメータの個数だけ存在します。

属性

```
<Parameter id="ErrorMsg" type="STRING">SUCC</ Parameter >
```

属性名	説明
id	パラメータ名
type	パラメータ値のタイプ

備考

nexacro platformでは次のような必須パラメータを使うように定義した。nexacro platformとの連動のためにはパラメータを設定しなければなりません。

```
<Parameter id="ErrorCode">1</Parameter>
<Parameter id="ErrorMsg">SUCC</Parameter>
<Parameter id="cachetype">Session</Parameter>
```

id	説明
ErrorCode	Error Code. -Transaction 関数呼び出しの時、0より小さい場合にはユーザが定義したエラーで、InputDatasetにUpdateStatusを反映せずに失敗処理されます。 0以上の場合には、ユーザが定義した正常状態で、InputDatasetにUpdateStatusを反映して成功処理されます。 -Load 関数呼び出しの時 0より小さい場合には、ユーザが定義したエラーで失敗処理されます。

id	説明
	0以上の場合には、ユーザが定義した正常状態で成功処理されます。 デフォルト値：0
ErrorMsg	Error Message 未指定時：ErrorCodeが未指定または0(デフォルト)の場合はSUCCESS、それ以外はFAILED
CacheType	Cache方法のTypeを指定する。Typeは Session、Dynamic、Static、Noneに区分されます。 デフォルト値：Session

typeに指定できるデータタイプは次の通りです。

- STRING: ストリング。Columnのsizeにかかわらず長さの制限がありません。(ただしsystemで一回に割り当て可能な大きさである2GBに制限される)
    - 0 length string の場合
- <Parameter id="ErrorMsg" type=”STRING”></Parameter>
    - Nullの場合
- <Parameter id="ErrorMsg" type=”STRING”/>と区分する。
- INT: 整数(-2^{31}~ 2^{31}-1)
- FLOAT、DECIMAL: (±2.2X10^{-308} ~±1.7X10^{308})
- BIGDECIMAL: 実数(±10^{-1056} ~ ±10^{1056})
- DATE: YYYYMMDD(-8192/01/01 ~ 8191/12/31)
- DATETIME：YYYYMMDDHHmmssuuu (msecを含めることが可能)
- TIME：6桁 HHmmssuuu(msecを含めることが可能)
- BLOB: バイナリデータ

3.2.4 Dataset

概要

一つのDataset内容の値を持っているエレメント。子ノードにはDatasetの構造を表現するためのColumnInfoとDatasetの実際のデータを持っている Rowsがあります。

子ノード

ColumnInfo, Rows

繰り返し指定

存在しないか、xmlが送信するDatasetの個数だけ存在します。

属性

```
<Dataset id="History">
```

属性名	説明
id	Dataset 名

3.2.5 Dataset > ColumnInfo

概要

Datasetのスキーマを表現するためのエレメント。実際にスキーマの情報は子ノードであるConstColumnとColumnが表現しColumnInfoはこれらを含む集合の役割を持ちます。固定値のカラムスキーマはConstColumnが非固定値のカラムスキーマはColumnが表現します。

子ノード

ConstColumn, Column

繰り返し指定

存在しないか、1つ存在する場合があります。

ColumnInfoの子要素は、ConstColumn、Column順に配置する必要があり、ConstColumnが常に最初に配置される必要があります。
ConstColumn、Columnが混ざって配置される場合には、アプリケーションからのデータを処理できない場合があります。

```
(X) ConstColumnが先に配置されない場合
<ColumnInfo>
    <Column id="stockCode" size="5" type="STRING"/>
    <Column id="currentprice" size="10" type="INT"/>
    <ConstColumn id="market" size="10" type="STRING" value="kse"/>
    <ConstColumn id="openprice" size="10" type="INT" value="15000"/>
</ColumnInfo>
...
```

```
(X) ConstColumn, Columnが混ざって配置された場合
<ColumnInfo>
    <ConstColumn id="market" size="10" type="STRING" value="kse"/>
    <Column id="currentprice" size="10" type="INT"/>
    <ConstColumn id="openprice" size="10" type="INT" value="15000"/>
    <Column id="stockCode" size="5" type="STRING"/>
</ColumnInfo>
...
```

3.2.6 Dataset > ColumnInfo > ConstColumn

概要

常に固定値を持つ Columnのスキーマを表現します。

子ノード

なし。

繰り返し指定

存在しないか、固定値カラムの数だけ存在します。

属性

```
<ConstColumn id="systemName"
     size="255"
     type="STRING"
     value="Production"/>
```

属性名	説明
id	カラム名
size	最大データサイズ
type	カラムのデータタイプ
value	カラムの固定値

備考

typeに指定できるデータタイプは次の通りです.

- STRING: ストリング。カラムのサイズにかかわらず長さの制限がありません。(ただしsystemで一回に割り当て可能な大きさである2GBに制限される)
- INT: 整数(-2^{31} ~ $2^{31}-1$)
- FLOAT、DECIMAL: ($\pm 2.2X10^{-308}$ ~$\pm 1.7X10^{308}$)
- BIGDECIMAL: 実数($\pm 10^{-1056}$ ~ $\pm 10^{1056}$)
- DATE: YYYYMMDD(-8192/01/01 ~ 8191/12/31)
- DATETIME : YYYYMMDDHHmmssuuu (msecを含んで可能)

- TIME：6桁 HHmmssuuu(msecを含めることが可能)
- BLOB: バイナリデータ

3.2.7 Dataset > ColumnInfo > Column

概要

パラメータの値を明示します。

子ノード

なし。

繰り返し指定

存在しないか、非固定値カラムの個数だけあります。

属性

```
<Column id="department"
     size="255"
     type="STRING"
     prop="text"
     sumtext="小計"/>
```

属性名	説明
id	カラム名
size	データの最大大きさ
type	カラムのデータタイプ
Prop	Summuryの時タイプを定義
Sumtext	porp値が textである時は文字列

備考

typeに設定される値は ColumnInfoでの typeと同じです。

Propは次のような値を持7.2.9ちます。

- count : このカラムのrow個数をsummary表示します。
- sum: このカラムのrow値の合計をsummaryに表示します。
- max: このカラムのrowの中でMax値をsummaryに表示します。

- min: このカラムのrowの中でMin値をsummaryに表示します。
- avg: このカラムのrow値の平均をsummaryに表示します。
- Text: summaryに sumtext属性値を表示します。

3.2.8 Dataset > Rows

概要

データセットの各Rowを内包するエレメント。実データであるRowを子ノードとして持ちます。

子ノード

Row

繰り返し指定

存在しないか、1つ存在する場合があります。

3.2.9 Dataset > Rows > Row

概要

データセットの一つのRowのデータを表現するためのエレメント。実際に各カラムの値を表現するColを子ノードとして有しています。 そしてtypeが"update"の場合、変更される前のRowの値を有するOrg_Rowが、子ノードとしてあります.

子ノード

Col, Org_Row

繰り返し指定

存在しないか、データの Row個数だけあります。

属性

```
<Row type="insert">
     <Col id="currentCode">10001</Col>
     <Col id="currentprice">13400</Col>
</Row>
<Rows>
     <Row type="update">
          <Col id="currentCode">10001</Col>
          <Col id="currentprice">13400</Col>
          <OrgRow>
               <Col id="currentCode">10001</Col>
               <Col id="currentprice">13700</Col>
          </OrgRow>
     </Row>
</Rows>
<Row type="delete">
     <Col id="currentCode">10001</Col>
     <Col id="currentprice">13400</Col>
</Row>
```

属性名	説明
type	insert：原本 Datasetに追加された Row.

属性名	説明
	update: 原本 Datasetに対して変更された Row。子ノードにOrg_Rowを含む。Org_Rowは変更前の原本Rowです。
	delete: 原本Datasetに削除されたRowを意味します.

備考

3.2.10 Dataset > Rows > Row > Col

概要

データセットの各カラム値を表現します。

子ノード

なし。

繰り返し指定

存在しないか、カラムの個数だけあります。

属性

```
<Col id="department">management</Col>
```

属性名	説明
Id	カラム名。ColumnInfo の子ノード(ConstColumn 、Column)で設定した名前と同じである。

備考

0 length string の場合

```
<Col id="department"></Col>
or
<Col id="department"/>
```

Nullの場合Tagがないと認識します。

3.2.11 Dataset > Rows > Row > OrgRow

概要

データセットのRow値が変更された時、元々の値を有するエレメント。子ノードとして実際の値を有するColがあります。

子ノード

Col

繰り返し指定

存在しないか、1つ存在する場合があります。

備考

親エレメントである Rowと似ていますが属性がありません。

3.2.12 Dataset > Rows > Row > OrgRow > Col

概要

変更される前のカラム値を持っています。

子ノード

なし

繰り返し指定

存在しないか、カラムの個数だけあります。

属性

```
<Col id="department">management</Col>
```

属性名	説明
id	カラム名。ColumnInfo の子エレメント(ConstColumn 、Column)で設定した名と同じ

属性名	説明
	です.

付録 A.

パスの設定

A.1　エイリアスパス

nexacro platformは使用者のPCのパスの設定に以下のエイリアスパスを使用します。

エイリアスパス名	実際パス名
%nexacro%	%PROGRAMFILES%\nexacro\14
%USERAPP%	%LOCALAPPDATA%LOW\nexacro\14 \data\data\[packageName]\files\nexacro\ (Android) \Library\Caches\nexacro\ (iOS)
%UPDATE%	%USERAPP%\Update
%CACHE%	%USERAPP%\Cache
%WEBDEPLOY SETUP%	Server Context root
%WEBDEPLOY PROJECT%	Server Context root
%WEBDEPLOY FRAMEWORK%	%WEBDEPLOY PROJECT%\nexacro14lib\framework
%WEBDEPLOY COMPONENT%	%WEBDEPLOY PROJECT%\nexacro14lib\component
%WEBDEPLOY RESOURCE%	%WEBDEPLOY PROJECT%\nexacro14lib\resources
%WEBDEPLOY THEME%	%WEBDEPLOY PROJECT%_theme_

このエイリアスパスは、配布情報のパスや応用プログラムでパス情報を入力する時に使用することができます。

実際のパス名は、製品バージョンや国別ライセンスポリシーに基づいて異なる場合があります。

A.2 nexacro.xml

nexacro.xmlファイルは、使用者別に情報を格納するために使用するxmlです。ファイルパスは以下の通りです。

%USERAPP%

A.3 相対パス

A.3.1 プロジェクト内のパスの相対パスサポート有無

下記の表はプロジェクト内でメインパスの種類及び相対パスのサポート有無、サービス利用パス(Service::a.xfdlの形式のパス)のサポート有無を整理したものです。

ファイル種類	項目	相対パス		サービス利用パスのサポート
		サポート	基準パス	
Typedefinition	Component UpdateURL ❶	あり	TypeDefintion	あり
	Serviceパス	あり	TypeDefintion	なし
GlobalVariable	Image	あり	GlobalVariable	あり
XADL	TypeDefinition ❷	あり	XADL	なし
	GlobalVariable	あり	XADL	あり
	Engineパス ❸	あり	XADL	なし
	Liceneseパス ❹	あり	XADL	なし
	Themeid	あり	XADL	あり

1. Component UpdateURL

   コンポーネントをアップデートするためのサーバーパスであり、ツールの[TypeDefinition → Update → UpdateUrl]を意味します。

   但し、このパスのドメイン(例：http://a.b.c/pathで a.b.c）がXADLのパスと一致しない場合はアップデートしません。もし、XADLがローカルパスである場合はドメイン情報と関係なくアップデートします。

2. TypeDefinition

   サービスのパスであり、ツールの[TypeDefinition→Services→url]を意味します。

3. Engineパス

   エンジンのアップデートのためのパスであり、ツールの[ADL → engineurl]を意味します。但し、XADLパスとドメインが一致しないとアップデートしません。 XADLがローカルパスの場合はドメインと関係なくアップデートします。

4. Liceneseパス

   TypeDefinitionが読まれる前にライセンスをチェックするためのパスです。

   ツールの[ADL → licenseurl]

   但し, XADLパスとドメインが一致する必要があります。XADLがローカルパスの場合はドメインと関係なくライセンスファイルをチェックします。

A.3.2 プラグイン方式

下記の表は、HTMLでnexacro platformをプラグイン方式に実行させる場合、HTMLに設定するパスの種類及び相対パスのサポート有無などを整理したものです。

区分(HTML)	項目	HTML 相対パスのサポート	XADL 相対パスのサポート	サービス利用パスのサポート
CODEBASE	SetupEngine.cab	関係なし	関係なし	関係なし
	nexacro14_Launcher.cab	関係なし	関係なし	関係なし
SetupEngine	Xadlpath プロパティ	あり	関係なし	関係なし
NexacroLauncherAX	Xadlpath プロパティ	あり	関係なし	関係なし
	Loadingimage プロパティ	あり	関係なし	関係なし
	Splashimage プロパティ	あり	関係なし	関係なし
	makeshortcut()	あり	関係なし	関係なし

- 4GLではサポートされず、HTMLでのみサポートされます。
- HTMLの相対パスの基準パスは、 HTMLのパスです。

付録 B.

例外状況

操作中に例外的に発生する状況についての対処方法をまとめたものです。使用環境によって適用する方法が異なる場合がございますので参考としてのみお使いください。

B.1　ウェブブラウザーオプション

ユーザーがウェブブラウザで、特定の目的のためにオプションを変更した場合に、アプリケーションで該当機能を使用している場合、正常な動作できない可能性があります。主に以下のような状況で問題が発生することがあります。

B.1.1　javascriptの活性化

ユーザーがウェブブラウザで、JavaScriptが使用できないように設定した場合、ファイルダウンロードが使用できないように制限した場合、アプリケーションが正常に実行されません。一般的に、デフォルトオプションは活性化状態ですが、ユーザーが別のオプションを選択することができますので、アプリケーションが画面に表示されない場合は、該当オプションを先に確認します。

使用するウェブブラウザによってJavaScriptの設定を活性化にする必要があります。使用しているバージョンによって、該当オプションは異なる可能性があり、詳細は、以下のリンクまたは各ウェブブラウザのヘルプを参照してください。

> http://www.enable-javascript.com

Internet Explorerの場合には、以下のオプションで変更します。

> Tools > Internet options > Security > Custom level > Internet Zone > Scripting
> > Active scripting

B.1.2 ファイルダウンロード活性化

ユーザーがウェブブラウザで、ファイルダウンロードが使用できないように制限した場合、Filddownloadコンポーネントを使用するファイルダウンロードができません。一般的に、デフォルトオプションは活性化状態ですが、ユーザーが別のオプションを選択することができますので、ファイルダウンロードが実行されない場合は、該当オプションを先に確認します。

使用するウェブブラウザによってJavaScriptの設定を活性化にする必要があります。使用しているバージョンによって、該当オプションは異なる可能性があり、詳細は、以下のリンクまたは各ウェブブラウザのヘルプを参照してください。

Internet Explorerの場合には、以下のオプションで変更します。

> Tools > Internet options > Security > Custom level > Internet Zone > Downloads
> > File Download

B.1.3 HTTP 1.1活性化

ユーザーがウェブブラウザで、 HTTP 1.1が使用できないように制限した場合、アプリケーションが正常に動作しません。一般的に、デフォルトオプションは活性化状態ですが、ユーザーが別のオプションを選択することができますので、アプリケーションが実行されない場合は、該当オプションを先に確認します。

Internet Explorerでのみ提供される機能であり、以下のオプションで変更します。

> Tools > Internet options > Advanced > HTTP settings > Use HTTP 1.1 through
> proxy connections
> Tools > Internet options > Advanced > HTTP settings > Use HTTP 1.1

> ウェブブラウザの設定に関係なく、ウェブサーバの設定でHTTP1.0を使用するように強制する場合、アプリケーションが遅くなったり、動作しない場合があります。すべてのユーザー環境が遅くなる場合は、ウェブサーバーの設定を確認してください。
> Apacheサーバの場合、`force-response-1.0` 設定が使用できます。
> http://httpd.apache.org/docs/2.2/en/env.html#special

B.1.4　XMLHTTP活性化

使用者がウェブブラウザでJavaScriptが使用できないように設定した場合には、アプリケーションが正常に動作しません。一般的に、デフォルトオプションは活性化状態ですが、ユーザーが別のオプションを選択することができますので、アプリケーションが画面に表示されない場合は、該当オプションを先に確認します。

Internet Explorerでのみ提供される機能であり、以下のオプションで変更します。

```
Tools > Internet options > Advanced > Security > Enable native XMLHTTP support
```

B.2 Internet Explorerの互換表示

Internet Explorerでサービスされているドメインをユーザーが互換表示対象に追加した場合には、画面が正常に出力されないことがあります。

nexacro platformアプリケーションの開発時に使用するHTMLファイルは、上位バージョンのIEの向上された性能を利用するために、各バージョンの標準モードで動作するようにMeta tagでレンダリングモードを指定し(IE= Edge)、該当モードでは、アドレス入力ウィンドウに互換表示アイコンが表示されません。

そのため、画面が割れるユーザーのブラウザの設定を確認して、標準モードで動作するようにガイドを提供したり、別のスクリプトで互換表示モードであるかどうかを確認して、ユーザーに通知することができます。

> 以下のコードは参考用です。開発環境に合わせて使用してください。
> 参考: https://msdn.microsoft.com/ja-jp/library/ms537503(v=vs.85).aspx

```
var agentStr = navigator.userAgent;
var mode;
if (agentStr.indexOf("Trident/6.0") > -1) {
     if (agentStr.indexOf("MSIE 7.0") > -1) {
          mode = "IE10 Compatibility View";
     } else {
          mode = "IE10";

}
else if (agentStr.indexOf("Trident/5.0") > -1) {
     if (agentStr.indexOf("MSIE 7.0") > -1) {
          mode = "IE9 Compatibility View";
     } else {
          mode = "IE9";

}
else if (agentStr.indexOf("Trident/4.0") > -1) {
     if (agentStr.indexOf("MSIE 7.0") > -1) {
          mode = "IE8 Compatibility View";
     } else {
          mode = "IE8";
     }
}
```

```
else {
     mode = "IE7";
}
document.title = "Browser Mode:\t" + mode;
```

B.3　既存のWeb画面にインラインフレームでコンテンツ追加

既に開発されて運営している画面にインラインフレーム(iframe)フォーマットでの nexacro platformコンテンツを追加する場合、インターネットエクスプローラから画面が出力されないことがあります。

nexacro platformアプリケーションの開発時に使用するHTMLファイルは、Edgeモードで動作するように構成されています。既存の画面に使用されたHTMLファイルにDTD(Document Type Definition)が指定されていない場合には、Quirksモード(Quirks mode)で動作しますが、この過程で、画面が正常に出力されないことがあります。

このような場合には、以下のようにメタタグを追加します。

```
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=Edge"/>
</head>
```

メタタグまたはHTTPヘッダーを使用した互換性の保証に関する内容は、以下のリンクを参照してください。
https://technet.microsoft.com/ja-JP/library/gg699448.aspx

付録 C.

nexacro platform X-APIインストール

nexacro platform X-APIはサーバーモジュールで提供され、サーバーとクライアント間でのデータ処理のために必要な機能を提供します。基本的なデータ送信、受信機能と簡単に必要なデータを処理する機能を提供してデータ処理を簡素化します。

> Dataset XML Formatに定義された内容によって動作するため、X-APIを使用せずに、別のサーバープログラムを作成して使用することもできます。

C.1　インストール

X-APIは、Javaベースのサーバモジュールで提供され、JDKまたはJRE1.4以降のバージョンが必要になります。

インストールCD内のnexacro-xapi/libディレクトリに含まれているjarファイルをWASの/WEB-INF/libディレクトリまたは定義されたクラスパスにコピーします。提供されるjarファイルは次のとおりです。

ファイル名	必須	説明	参照
nexacro-xapi-x.x.x.jar	Y	X-API	
commons-logging-x.x.x.jar	Y	X-API内部ロギング	Apache Commons Logging

C.2　ライセンス認証

ライセンスファイル(nexacro14_server_license.xml)をjarファイルと同じディレクトリ、または定義されたクラスパスにコピーします。

> 1つ以上のライセンスファイルが別のパスにコピーされた場合には、jarファイルと同じディレクトリにあるライセンスファイルを最初に適用します。

> サーバー環境設定によって、jarファイルとライセンスファイルをコピーし、WASの再起動が必要になる場合があります。

C.3　インストール確認

X-APIが正常にインストールされたかどうかは、以下のサンプルJSPファイルを作成して簡単に確認することができます。

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
    <head>
    <title>JarInfo</title>
    <style>
        * { font-family: Verdana }
    </style>
    </head>
    <body>
        <pre>
<%
    new com.nexacro.xapi.util.JarInfo().info(out);
%>
        </pre>
    </body>
```

```
</html>
```

正常にインストールがされた場合X-APIのバージョン情報と製品ライセンス情報が出力されます。

```
--- Manifest ---
Manifest-Version: 1.0
Built-By: Server Team
Created-By: 1.4.2_19-b04 (Sun Microsystems Inc.)
Ant-Version: Apache Ant 1.7.1
Main-Class: com.nexacro.xapi.util.JarInfo
Built-Date: April 23 2014

Name: com.nexacro.xapi
Specification-Title: X-API
Implementation-Title: X-API
Specification-Version: 1.0
Specification-Vendor: [illegible]
Implementation-Vendor-Id: com.nexacro
Implementation-Version: 1.0.0.4017
Implementation-Vendor: [illegible]

--- Log ---
INFO Loaded license file in JAR dir: path=D:\workspace\nexacro\WebContent\WEB-INF\lib\nexacro14_server_lic

--- License ---
product.name: nexacro platform
product.version: 14
product.function: Windows, WindowsRT, MAC, iOS, Android
customer.name: [illegible]
server.cpuCoreCount: [illegible]
server.ipAddress: [illegible]
date.activation: 2014-04-01
date.term: 2
date.term.unit: month
```

付録 D.

nexacro-xeniインストール

nexacro-xeniは、サーバーモジュールで提供され、クライアントからのファイル形式でデータを処理する際に必要な機能を提供します。ファイルに含まれているデータをグリッドコンポーネントにインポートするか、グリッドに連結されたデータをファイルにエクスポートする機能を処理します。

nexacro-xeniは次のようなファイル形式と機能をサポートします。

ファイル形式	拡張子	機能
Microsoft Excel 97-2003	xls	export / import
Microsoft Excel 2007/2010	xlsx	export / import
CSV format file	csv	import

D.1　インストール

nexacro-xeniは、Javaベースのサーバーモジュールで提供され、JDKまたはJRE1.8以降のバージョンが必要となります。

指定されたwar(Web application ARchive)ファイルを直接配置したり、warファイルを解凍して、必要なファイルをWASの /WEB-INF/libディレクトリまたは定義されたクラスパスにコピーして使用することができます。

> warファイルを直接配置せずに解凍してコピーする場合には、ライブラリのバージョンの違いのため問題が発生することがあります。

nexacro-xeniはApache POIライブラリを使用します。詳細 Apache POIの説明は以下のリンクを参照してください。
https://poi.apache.org/

nexacro-xeniが動作するには、X-APIがインストールされている必要があります。nexacro platform X-APIインストール 項目を参照してください。

既存のコンテキストにファイルをコピーした場合には、以下の内容をweb.xmlファイルに追加する必要があります。

```
<servlet>
    <servlet-name>XExportImport</servlet-name>
    <servlet-class>com.nexacro.xeni.services.GridExportImportServlet</servlet-
class>
</servlet>
<servlet>
    <servlet-name>XImport</servlet-name>
    <servlet-class>com.nexacro.xeni.services.GridExportImportServlet</servlet-
class>
</servlet>

<servlet-mapping>
    <servlet-name>XExportImport</servlet-name>
    <url-pattern>/XExportImport</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>XImport</servlet-name>
    <url-pattern>/XImport</url-pattern>
</servlet-mapping>

<context-param>
    <param-name>export-path</param-name>
    <param-value>/export</param-value>
</context-param>
<context-param>
    <param-name>import-path</param-name>
    <param-value>/import</param-value>
</context-param>
```

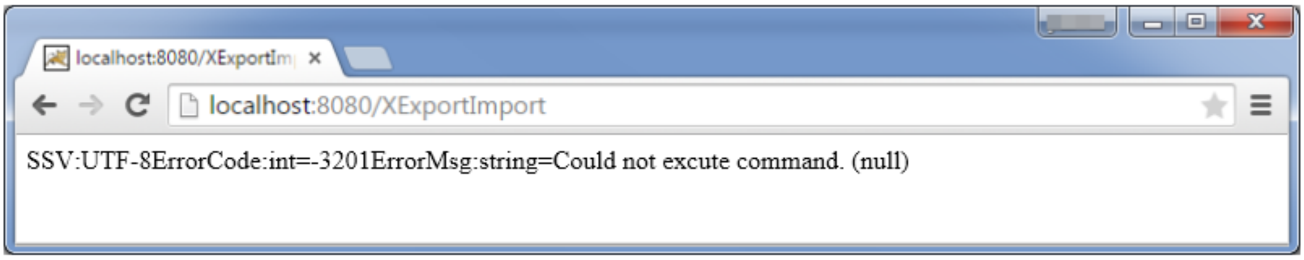
```
<context-param>
    <param-name>monitor-enabled</param-name>
    <param-value>true</param-value>
</context-param>
<context-param>
    <param-name>monitor-cycle-time</param-name>
    <param-value>30</param-value>
</context-param>
<context-param>
    <param-name>file-storage-time</param-name>
    <param-value>10</param-value>
</context-param>
```

> jarファイルのリストと web.xmlに追加された内容は、nexacro-xeniバージョン1.1基準です。更新に基づいて、リストが変更されることがあります。

D.2　インストールの確認

通常の配置またはコピーされた場合、次のURLをWebブラウザで開いて、メッセージが正常に表示されることを確認します。

```
http://127.0.0.1:8080/XExportImport
```



> URLは、インストールパスによって異なる場合があります。warファイルを直接配置した場合、URLは下記の通りです。
> `http://127.0.0.1:8080/nexacro-xeni-N/XExportImport`

D.3　主な設定

サーブレットコンテキストにパラメータで設定された主な項目は以下の通りです。

項目名	デフォルト値	説明
export-path	/export	ダウンロードするファイルを一時的に保存するパスを指定します。
import-path	/import	グリッドコンポーネントにインポートファイルを一時的に保存するパスを指定します。 monitor-enabled設定値がtrueである場合には、データ処理後すぐに削除します。
monitor-enabled	true	ファイルマネージャを実行するかどうかを指定します。 falseに指定した場合には、ファイルマネージャが動作せず、一時ファイルを削除しません。
monitor-cycle-time	30	一時保存されたダウンロードするファイルを削除する周期(サイクル)を「分」単位で設定します。 該当周期にファイルマネージャを実行して、削除するファイルをチェックし、削除処理します。
file-stroage-time	10	一時的に保存されたダウンロードするファイルを保存する時間を「分」単位で設定します。 ファイルマネージャの実行時に生成した後、指定した時間が経過したファイルは削除されます。

詳しくは、配布される warファイルに含まれている readme.txtファイルを参照してください。

D.4　例題

単純なデータ処理の例を説明します。詳細なプロパティやメソッド、イベントは、ヘルプで以下の項目を参照してください。

`Objects > Misc. Objects > ExcelExportObject`

D.4.1 Export

Gridコンポーネントにバインドされたデータをサーバーに送信してファイルで生成し、生成されたファイルをダウンロードします。次の例では、Grid00コンポーネントにバインドされたデータをファイルで生成してダウンロードします。

```
this.Button00_onclick = function(obj:Button,  e:nexacro.ClickEventInfo)
{
    this.exportObj = new ExcelExportObject();
    var ret = this.exportObj.addExportItem(nexacro.ExportItemTypes.GRID,
        this.Grid00, "Sheet1!A1");
    this.exportObj.set_exporturl("http://localhost:8080/XExportImport");
    this.exportObj.exportData();
}
```

D.4.2 Import

ユーザーが持っているファイルをサーバーに送信して、データセットに変換します。ユーザーは、変換されたデータセットをダウンロードして業務を処理します。

```
this.Button00_onclick = function(obj:Button,  e:nexacro.ClickEventInfo)
{
    this.importObj = new ExcelImportObject("Import00",this);
    this.importObj.addEventHandler("onsuccess", this.Import_onsuccess, this);
    this.importObj.set_importurl("http://localhost:8080/XExportImport");
    this.importObj.importData("","output=ds","Dataset00=ds");
}
this.Import_onsuccess = function(obj,e)
{
    this.Grid00.createFormat();
}
```